SECRECY AND ANONYMITY IN INTERACTIVE SYSTEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Kevin Ross O'Neill

August 2006

SECRECY AND ANONYMITY IN INTERACTIVE SYSTEMS

Kevin Ross O'Neill, Ph.D.

Cornell University 2006

When building systems that guarantee confidentiality, system designers must first define confidentiality appropriately. Although researchers have proposed definitions of properties such as secrecy, anonymity, and privacy for a wide variety of system models, general definitions that are intuitive, widely applicable, and sufficiently formal have proven surprisingly elusive. The goal of this dissertation is to provide such a framework for systems that interact with multiple agents, emphasizing definitions of secrecy (to rule out unwanted information flows) and anonymity (to prevent observers from learning the identity of an agent who performs some action). The definitions of secrecy extend earlier definitions of secrecy and nondeducibility given by Shannon and Sutherland. Roughly speaking, one agent maintains secrecy with respect to another if the second agent cannot rule out any possibilities for the behavior or state of the first agent. These definitions are characterized syntactically, using a modal logic of knowledge. Definitions of anonymity are given, with respect to agents, actions, and observers, and are also stated in terms of a modal logic of knowledge. The general framework is shown to handle probability and nondeterminism cleanly, and to be useful for reasoning about asynchronous systems as well as synchronous systems. It also suggests generalizations of secrecy and anonymity that may be useful for dealing with issues such as resource-bounded reasoning. Finally, the dissertation leverages these def-

initions of secrecy and formulates new strategy-based information-flow conditions for a simple imperative programming language that includes input and output operators. A soundness theorem demonstrates the feasibility of statically enforcing the security conditions via a simple type system.

BIOGRAPHICAL SKETCH

Kevin O'Neill was born in Kelowna, British Columbia in 1977. His interest in computers began early. In kindergarten, he appeared on the front page of the local newspaper when a computer was loaned to his class for a day and he was photographed using it. When he received a 128-Kb Tandy computer in the fourth grade, which featured neither a disk drive nor pre-stored programs of any kind, he was forced to write new BASIC programs every time he turned on the machine. (Mostly they were copied from books, but he did once write a program to output the numbers from 1 to one million. It took a long time to execute.)

In his early years, Kevin wanted to be, variously, a magician, a puppeteer, a lawyer, an engineer, and a sound-recording expert. He spent much time building things with lego, and he occasionally liked to read the encyclopedia for fun. He played piano and was fond of classical music, but had lousy practice habits. Rachmaninoff's preludes were his downfall.

After reading a book by Oliver Sacks in high school, Kevin decided that he wanted to be a neurologist. In his first year at the University of British Columbia, however, he discovered that he was bad at memorizing things but pretty good at writing recursive functions. When he almost fainted while observing rat surgery the following year he decided it was time to switch majors.

As a computer science student, Kevin was fortunate to work with professors in the Laboratory for Computational Intelligence at UBC, where he got his first exposure to research. After a summer project he decided he wanted to be a computer scientist. Lazily, he applied only to two Ph.D. programs; luckily, they both accepted him. He moved to Ithaca in August, 2000 to start graduate work at Cornell. He maintains that the following six years were the best and worst years

of his life. (Notably, they coincided with the administration of George W. Bush.) Highlights of his time in Ithaca include nights at the Chapter House Pub, cycling on White Church Road, the Great Fire of 2005, Beaujolais Nouveau Day 2003, and, best of all, burgers and beer at Rogue's Harbor after afternoons on Cayuga Lake.

Kevin is addicted to the New Yorker magazine, frequently annoyed by the New York Times, and enthusastic about live jazz, sailing, and weekends in big cities. He still hasn't managed to learn how to whistle.

For my parents, Pat and Michelene.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Joe Halpern, for his academic guidance and generous financial support throughout my graduate studies at Cornell. Through years of helpful criticism, he helped me to learn how to recognize and strive towards excellent research and writing. I'm still consistently impressed by the speed and quality of his feedback about my work, glad to have had the experience of coauthoring papers with him, and grateful for the research assistantships and travel support that he has provided.

I would also like to thank the other members of my committee, Andrew Myers and Eric Friedman, and also Fred B. Schneider, for their oversight of my work and their suggestions for research directions and improvements to the papers that this dissertation comprises. Andrew also provided valuable and thorough suggestions on how to improve the dissertation itself.

The work in Chapter 6 was joint work with Michael Clarkson and Stephen Chong. I am grateful to both of them for making the last year of my graduate work rewarding and fun and for helping me do the work necessary to finish my Ph.D.

My graduate work was supported in part by a fellowship from the National Science and Engineering Research Council of Canada, and I am grateful for their financial support—and to everyone who wrote letters of recommendation during my multiple applications. I am also grateful for financial support from NSF under grants IRI-96-25901, IIS-0090145, CTC-0208535, and 0430161; by ONR under grants N00014-00-1-03-41 and N00014-01-10-511; by AFOSR under grant F49620-02-1-010; and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grants N00014-01-1-0795 and FA9550-

05-1-0055.

I'd like to thank Claire Cardie, David Skalak, and Éva Tardos for personal and professional advice at various stages of my graduate career. And I'd also like to sincerely thank everyone in the Department of Computer Science, notably Eric Breck and Cindy Robinson, who were so generous with their support following my unfortunate house fire in 2005.

On a more personal level, my parents deserve thanks for their early sacrifices, paying for private school, piano lessons, and computers when they could least afford it. They also gave me the support and freedom to do whatever I wanted with my life and never doubted my ability to do it. I'm also grateful to my sister, Kristy, and to the rest of my extended family, all of whom cheered me on throughout college and graduate school.

Colleagues and friends at Cornell greatly improved the quality of my life in Ithaca. Dan Grossman was my apartmentmate and informal mentor during my first three years and, in addition to being a great friend, gave me countless tips on navigating the world of academic computer science. Riccardo Pucella, Vicky Weissman, and Sabina Petride were outstanding academic "siblings" and generously provided helpful feedback on my papers and talks. I also had a succession of great officemates, including Cristian Bucila, Adina Crăiniceanu, Junhwan Kim, Niranjan Nagarajan, Amy Gale, and Michael George.

Other friends—including (but not limited to) Alice Te Punge Somerville, Amy Gale, Ariane Kissam, Casey Westerman, Christina Dunbar-Hester, Cyrus Mody, Dana Brown, Daniel Marques, Gavin Hurley, James Slezak, Jay Schweig, John Downer, Stephen Chong, Rob Abramovitch, Tim Roughgarden, Zorka Milin, and everyone at the Stewart Little Co-op—contributed significantly toward my happi-

ness and well-being throughout my tenure in Ithaca.

Finally, I'm grateful to old friends outside Ithaca, including Callum Campbell, Darcy Nebergall, David Treleaven, Jesse Jackson, Jonathan Schreiber, and Kim de Simone, all of whom provided valuable friendship and the occasional futon to crash on.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The goal of this dissertation is to provide a formal framework for reasoning about confidentiality properties in systems with which multiple agents interact over time. The importance of confidentiality in multiagent systems has increased greatly during the past several years due to the widespread use of communication networks such as the Internet. Computer users may be reluctant to engage in useful activities such as Web browsing, message sending, and file sharing unless they can receive guarantees that their privacy or anonymity will be protected to some reasonable degree. Similarly, large organizations want guarantees that their communication systems will not leak confidential data to unauthorized users.

When building systems that provide confidentiality guarantees, system designers must first start with appropriate definitions of confidentiality. Though definitions of properties such as secrecy, anonymity, and privacy have been proposed for a wide variety of system models, general definitions that are intuitive, widely applicable, and sufficiently formal have proven surprisingly elusive.

Properties such as secrecy, anonymity, noninterference, privacy, and so on can be construed as providing answers to the following set of questions:

- What information needs to be hidden?
- Who does it need to be hidden from?
- How well does it need to be hidden?

By analyzing confidentiality properties with these questions in mind, it often becomes clear how different properties relate to each other. These questions can also

serve as a test of a definition's usefulness: a property should be able to provide clear answers to these three questions.

We focus here on two particular confidentiality properties: *secrecy*, which requires that the state of one agent remains hidden from others, and *anonymity*, which requires that the identity of the agent who performs some action remains hidden from other observers. Our definitions of secrecy and anonymity focus on the knowledge of agents who interact with a system. Accordingly, we state many of our definitions in terms of a logic of knowledge. By formalizing secrecy and anonymity in terms of knowledge we can capture the intuitions that practitioners have and also equate our knowledge-based definitions with more standard semantic definitions.

The use of the term "secrecy" in this dissertation is perhaps somewhat nonstandard. In contrast to definitions that are concerned with the secrecy of particular secrets (such as encrypted messages), the definitions of secrecy presented here aim to ensure that one agent learns nothing at all about the state of another. The term "total secrecy" might be more indicative of the general class of properties in which we are interested, but we avoid using it as an umbrella term both for brevity and because we use the term in a specific technical sense later in the dissertation. Our definitions of secrecy can be viewed as restrictions on the information that may flow from one agent to another, and are similar in spirit to other information-flow properties such as noninterference.

We believe that it is important to provide broad, general definitions of secrecy and anonymity for a general class of multiagent systems—definitions that emphasize the underlying unity of the notions we are trying to capture. Our work is intended to to do just that. Although our definitions should be appropriate for

a wide variety of settings, we pay special attention to the domain of imperative programming languages. In particular, we apply our definitions of secrecy to formalizing definitions of information-flow security for interactive programs written in a simple imperative language. We hope it will be clear that our definitions apply equally well to other settings.

In the remainder of this introductory chapter we motivate the need for a new framework in which we can couch semantic definitions of secrecy and anonymity. In Section 1.1 we discuss secrecy and in Section 1.2 we discuss anonymity. In Section 1.3 we discuss definitions of information-flow security for imperative programs, and in Section 1.4 we describe the structure of the rest of the dissertation.

1.1 Secrecy and information flow

In the past three decades there have been many attempts to define what it means for a system to be perfectly secure, in the sense that one group of agents is unable to deduce anything at all about the behavior of another group. More generally, many papers in computer science have, in a variety of different settings, defined properties of secrecy and privacy and have discussed techniques for achieving these properties. In the computer-security literature, early definitions of "perfect security" were based on two different intuitions. *Noninterference* [31] attempted to capture the intuition that an agent at a high security level is unable to interfere with an agent at a lower security level, while *nondeducibility* [87] attempted to capture the intuition that an agent at a low security level is unable to deduce anything about the state of agents at a higher security level. Others definitions have involved a notion of *information flow*, and taken a system to be secure if it is impossible for information to flow from a high user to a low user. With these basic ideas

in mind, definitions of security have been provided for a wide variety of system models, including semantic models that encode all possible input/output behaviors of a computing system and language-based models that deal with process algebras and with more traditional constructs such as imperative programming languages. (Focardi and Gorrieri [22] provide a classification of security properties expressed using process algebras; Sabelfeld and Myers [76] give a survey of language-based techniques.)

Sutherland's definition of nondeducibility was based on a simple idea: a system can be described as a set of "worlds" that encode the local states of users, and security is maintained if high and low states are independent in the sense that a low user can never totally rule out any high state based on his own local state. As we shall see, nondeducibility is closely related to Shannon's [81] probabilistic definition of secrecy in the context of cryptography, which requires high and low events to be probabilistically independent. This can be shown to imply that the low agent's posterior probability of a high event should be the same as his prior probability of that event before he began interacting with the system. (Nondeducibility is also closely related to Cohen's earlier definition of *strong dependency* [7]. Cohen's work is concerned with information transmission over channels, which are represented as functions that transform inputs into outputs, whereas Sutherland's work is concerned with possible worlds to represent states of a system, but their definitions are essentially identical.)

Definitions of noninterference that follow Goguen and Meseguer's early work (see, for example, McCullough [56] and McLean [58]) are quite different in flavor from the definitions of Shannon and Sutherland. Typically, they represent the system as a set of input/output traces, and deem the system secure if the set

of traces is closed under operations that add or remove high events. Variants of this idea have been proposed to deal with issues such as verification, system composition, timing attacks, and so on. Although these definitions have been useful for solving a variety of technical problems, the complexity of some of this work has, in our view, obscured the simplicity of earlier definitions based on the notion of independence. While nondeducibility has been criticized for its inability to deal with a variety of security concerns, we claim that the basic idea captures notions of secrecy and privacy in an elegant and useful way.

We define secrecy in terms of an agent's knowledge, using the "runs-and-systems" framework [20], which generalizes the standard input/output trace models that have been used in many definitions of noninterference. The trace-based approach has been concerned primarily with the input and output values exchanged as a user or observer interacts with the system. Thus, with a trace-based approach, it is possible to define secrecy only for systems that can be characterized by observable input and output events. This is insufficient for modeling a variety of interesting systems. As Focardi and Gorrieri [22] point out, for example, it is difficult to deal with issues such as deadlock using a purely trace-based approach. It is also difficult to represent an agent's notion of time in systems that may exhibit differing degrees of synchrony. As we shall see, the added generality of the runs-and-systems approach lets us deal with these issues in a straightforward way.

Many frameworks for reasoning about information flow have assumed, often implicitly, a very coarse notion of uncertainty. Either an agent knows, with certainty, that some fact is true, or she does not; a definition of secrecy therefore amounts to a characterization of which facts some agent must not know, or which facts she must think are possible. Indeed, this is the intuition that we make precise

in Section 3.1.4. In the literature, such definitions are called *possibilistic*, because they consider only what agents consider possible or impossible. In practice, however, such a coarse-grained notion of uncertainty is simply too weak. It is easy to concoct examples where one agent has possibilistic secrecy but where intuition suggests that secrecy is not maintained. We extend our definitions of secrecy to incorporate probability, a much more fine-grained notion of uncertainty, using a standard approach for reasoning about probability in the runs-and-systems framework [44]. Just as Shannon's definitions of secrecy can be viewed as a probabilistic strengthening of Sutherland's definition of nondeducibility, our definitions of probabilistic secrecy generalize the possibilistic definitions we give. In fact, there is a sense in which they are the same definitions, except with a different measure of uncertainty—a point made precise when we generalize them using *plausibilistic measures* in Section 3.3.

We also provide syntactic characterizations of secrecy, using a logic that includes modal operators for reasoning about knowledge and probability. We discuss what it means for a fact to "depend on" the state of an agent and show that secrecy can be characterized as the requirement that low agents never know any fact that depends on the state of a high agent. (In the probabilistic case, the requirement is that low agents must think that any such fact is equally likely at all points of the system.) This knowledge-based characterization lets us make precise the connection between secrecy (of one agent with respect to another) and the notion of a "secret," that is, a fact about the system that an agent is not allowed to know. This syntactic approach also opens the door to natural generalizations of information-flow properties that require secrecy for only some facts, and allows us to consider notions of secrecy based on more computational notions of knowledge,

which may be more appropriate for resource-bounded agents.

As we show in Chapter 5, our approach provides insight into a number of other information-flow conditions that have been proposed in the literature. We illustrate this point by considering *separability* [58], *generalized noninterference* [58], *nondeducibility on strategies* [93], and *probabilistic noninterference* [32]. One of our goals in this chapter, obviously, is to convince the reader that our definitions are in fact as general as we claim they are. More importantly, we hope that providing a unified framework for comparing definitions of secrecy will facilitate the cross-fertilization of ideas.

1.2 Anonymity

A variety of systems have been built to provide anonymity for the senders and receivers of electronic communication. (See, for example, [1, 30, 51, 71, 82, 88].) The goal of these systems, stated broadly, is to ensure that the identity of agents who transmit communication remains hidden from other users. Because these systems offer quite different guarantees about the degree of anonymity that they provide, we believe that it is helpful to have a formal framework that can be used to specify anonymity requirements and compare different systems. To this end, we provide a variety of definitions that formalize what it means for the identity of an agent who performs an action to be hidden from observers of the system.

Anonymity is obviously different from secrecy. Roughly speaking, a high agent maintains secrecy with respect to a low agent if the low agent never knows anything about the high user that he didn't initially know. In contrast, our definitions of anonymity say that an agent performing an action maintains anonymity with respect to an observer if the observer never learns certain facts having to do with

whether or not the agent performed the action. It is possible for an agent to have complete secrecy—for some definition of secrecy—while still not having very strong guarantees of anonymity. Conversely, it is possible to have anonymity without preserving secrecy. Thinking carefully about the relationship between secrecy and anonymity suggests new and interesting ways of thinking about anonymity. In addition, formalizing anonymity in terms of knowledge is useful for capturing the intuitions that practitioners have. Not surprisingly, our formalization of anonymity is similar in spirit to our formalization of secrecy.

We remark that we are not the first to use knowledge and belief to formalize notions of anonymity and other similar properties. Glasgow, MacEwen, and Panangaden [29] describe a logic for reasoning about security that includes both *epistemic* operators (for reasoning about knowledge) and *deontic* operators (for reasoning about permission and obligation). They characterize some security policies in terms of the facts that an agent is permitted to know. Intuitively, everything that an agent is not permitted to know must remain hidden. Our approach is similar, except that we specify the formulas that an agent is *not* allowed to know, rather than the formulas she is permitted to know. One advantage of accentuating the negative is that we do not need to use deontic operators in our logic.

Syverson and Stubblebine [89] use an epistemic logic to formalize definitions of anonymity, but the goal of our work is quite different from theirs. Syverson and Stubblebine focus on describing an axiom system that is useful for reasoning about real-world systems, and on how to reason about and compose parts of the system into adversaries and honest agents. Our focus, on the other hand, is on giving a semantic characterization of anonymity in a framework that lends itself well to modeling systems.

Shmatikov and Hughes [47] (whose work we discuss in more detail in Section 5.2.3) position their approach to anonymity as an attempt to provide an interface between logic-based approaches, which they claim are good for specifying confidentiality properties, and formalisms like CSP, which they claim are good for specifying systems. We agree with their claim that logic-based approaches are good for specifying properties of systems. But given an appropriate semantics for the logic, no such interface is necessary. There are many ways of specifying systems, but many end up identifying a system with a set of runs or traces and can thus be embedded in the runs and systems framework that we use.

As with secrecy, many definitions of anonymity do not deal with probability. Certainly, if an agent j believes that any of 1000 users (including i) could have performed the action that i in fact performed, then i has some degree of anonymity with respect to j. However, if j believes that the probability that i performed the action is .99, the possibilistic assurance of anonymity may provide little comfort. Most previous formalizations of anonymity have not accounted for probability. One significant advantage of our formalism is that it is completely straightforward to add probability by following the same approach that we use for secrecy. As we show in Section 4.2, this lets us formalize the (somewhat less formal) definitions of probabilistic anonymity given by Reiter and Rubin [71].

In this dissertation we are more concerned with defining and specifying anonymity properties than with describing systems for achieving anonymity or with verifying anonymity properties. We define what anonymity means by using syntactic statements with a well-defined semantics. Our work is similar in spirit to previous papers that have given definitions of anonymity, such as the proposal for terminology given by Pfitzmann and Köhntopp [68] and the information-theoretic

definitions of anonymity given by Diaz, Seys, Claessens, and Preneel [15].

1.3 Information-flow security for imperative programs

Secure programs should maintain the secrecy of confidential information. For sequential imperative programming languages, this principle has led to a variety of information-flow security conditions which assume that all confidential information is supplied as the initial values of a set of program variables. This assumption reflects an idealized *batch-job* model of input and output, whereby all inputs are obtained (as initial values of program variables) from users before the program begins execution, and all outputs are provided (as final values of program variables) after program termination. Accordingly, these security conditions aim to protect the secrecy only of initial values.

Many real-world programs, however, are interactive, sending output to and receiving input from their external environment throughout execution. Examples of such programs include web servers, GUI applications, and some command-line applications. The batch-job model is unable to capture the behavior of interactive programs because of dependencies between inputs and outputs. For example, a program implementing a challenge/response protocol must first output a challenge to the user and then accept the user's response as input; clearly, the user cannot supply the response as the initial value of a program variable. In contrast, the interactive model generalizes the batch-job model: any batch-job program can be simulated by an interactive program that reads the initial values of all relevant variables, executes the corresponding batch-job program, and finally outputs the values of all variables.

Given the prevalence of interactive programs, it is important to be able to

reason about their security properties. Traditionally, researchers have reasoned about information flow in interactive systems by encoding them as state machines (e.g., Mantel [54] and McLean [57, 58]) or as concurrent processes (e.g., Focardi and Gorrieri [22]) and applying trace-based information-flow security conditions. But since implementors usually create imperative programs, not abstract models, a need exists for tools that enable direct reasoning about the security of such programs.

We address that need by developing a model for reasoning about the information-flow security of interactive imperative programs. We give novel strategy-based semantic security conditions that are special cases of our definitions of secrecy. Our model achieves a clean separation of user behavior from program code by employing *user strategies*, which describe how agents interact with their environment. (The definitions build on the work of Wittbold and Johnson [93] and of Gray and Syverson [32], both of whom consider user strategies.) Moreover, our framework for secrecy suggests natural ways to give definitions of information-flow security for imperative programs that account for nondeterminism and randomization.

We also leverage previous work on static analysis techniques by adapting the type system of Volpano, Smith, and Irvine [92] to an interactive setting. Our results demonstrate the feasibility of enforcing our stringent definitions of security.

1.4 Overview

The material of Chapter 2 is a review of the runs-and-systems framework of Fagin, Halpern, Moses, and Vardi [20]. We describe their state-based approach for representing the local states of different agents in a multiagent system and an epistemic logic for reasoning about the knowledge of such agents. We also describe how

to add probability to the runs-and-systems framework, following the approach of Halpern and Tuttle [44].

Chapter 3 presents definitions of secrecy in the context of the runs-and-systems framework. Most of the definitions are semantic in nature; that is, they are concerned with the local states of agents. We do, however, provide several syntactic characterizations using the epistemic logic of Chapter 2, and we prove several results establishing equivalences between the semantic and logic-based definitions. We also give probabilistic definitions of secrecy in Chapter 3 and generalize many of the definitions using plausibility measures.

In Chapter 4 we give a variety of definitions of anonymity, most of which are stated syntactically (as logical formulas that are valid in systems that preserve anonymity).

Chapter 5 considers related work. We look at a number of definitions of secrecy, noninterference, and anonymity, and establish that some of those definitions can be cast as special cases of our definitions.

Chapter 6 considers definitions of secrecy for interactive imperative programs. We give an operational semantics for a small interactive language, provide definitions of noninterference that account for nondeterminism and randomization, and describe a type system for the language that is sound with respect to the definitions of noninterference.

The material in Chapters 3, 4, and 5 is joint work with Joseph Halpern and appears in [39], [40], and [41]. The material in Chapter 6 is joint work with Michael Clarkson and Stephen Chong [67].

Chapter 2

A Model for Multiagent Systems

This chapter provides a review of background material that is necessary for subsequent chapters. The technical material is not new: the framework for multiagent systems described here was established in work by Halpern and Fagin [37] and Halpern and Moses [38]. (See [20] for an excellent introduction to knowledge in multiagent systems.) The approach for reasoning about probability in multiagent systems described in Section 2.2 originates with Halpern and Tuttle [44].

2.1 Knowledge and multiagent systems

A multiagent system consists of n agents, each of whom is in some *local state* at a given point in time. We assume that an agent's local state encapsulates all the information to which she has access. In a security setting the local state of an agent might include initial information regarding keys, the messages she has sent and received, and perhaps the reading of a clock. The basic framework makes no assumptions about the precise nature of the local state.

We can view the whole system as being in some *global state*, which is a tuple consisting of the local state of each agent and the state of the environment, where the environment consists of everything relevant to the system that is not contained in the state of the agents. Thus, a global state has the form $(s_e, s_1, \ldots, s_n)$, where s_e is the state of the environment and s_i is agent i's state, for $i = 1, \ldots, n$.

A *run* is a function from time to global states. Intuitively, a run is a complete description of what happens over time in one possible execution of the system. A *point* is a pair (r, m) consisting of a run r and a time m. For simplicity, we take

time to range over the natural numbers. At a point (r, m), the system is in some global state $r(m)$. If $r(m) = (s_e, s_1, \ldots, s_n)$, then we take $r_i(m)$ to be s_i, agent i's local state at the point (r, m). Formally, a *system* consists of a set of runs (or executions). Let $\mathcal{PT}(\mathcal{R})$ denote the points in a system $\mathcal{R}$.

Given a system $\mathcal{R}$, let $\mathcal{K}_i(r, m)$ be the set of points in $\mathcal{PT}(\mathcal{R})$ that i thinks are possible at (r, m); that is,

$$\mathcal{K}_i(r, m) \triangleq \{(r', m') \in \mathcal{PT}(\mathcal{R}) : r'_i(m') = r_i(m)\}.$$

The set $\mathcal{K}_i(r, m)$ is often called an *i-information set* because, intuitively, it corresponds to the system-dependent information encoded in i's local state at the point (r, m).

A natural question to ask is where these runs come from. While the framework itself does not deal with this issue, in practice we are interested in systems where the runs are generated by a simple set of rules, such as a communication or security protocol, a program written in some programming language, or a process described in a concurrent process language. Translating such rules to a set of runs is not always straightforward, but doing so is often useful inasmuch as it forces us to think carefully about what features of the system are essential and relevant to the safety or correctness issues that we are interested in. With respect to secrecy, the way in which we model the local states of various agents is especially important. In particular, if we do not want the actions or choices made by one agent to affect the state of another agent, we should include sufficient information in the local state of the first agent to reason about those actions or choices. (This issue is discussed in more detail in Sections 3.2.4 and 5.1.2.)

To reason formally about secrecy and anonymity in multiagent systems, we use a logic of knowledge and time. Starting with a set Φ of primitive propositions,

we close off under negation, conjunction, the modal operators K_i for $i = 1, \ldots, n$, and $\Diamond$. In the context of security protocols, the set Φ might consist of primitive propositions corresponding to facts such as "the key is n" or "agent A sent the message m to B." As usual, $K_i\phi$ means that agent i knows ϕ; $K_i\phi$ at a point (r, m) if ϕ is true at all points in $\mathcal{K}_i(r, m)$. Finally, $\Diamond\phi$ is true at a point (r, m) if ϕ is true at some point on run r (either before, at, or after time m). While it is, of course, possible to define other temporal operators, the $\Diamond$ operator will prove particularly useful in our definitions.

We use the standard approach [20] to give semantics to this language. An interpreted system $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system and π is an interpretation for the primitive propositions in Φ that assigns truth values to the primitive propositions at the global states. Thus, for every $p \in \Phi$ and global state s that arises in $\mathcal{R}$, we have $(\pi(s))(p) \in \{\mathbf{true}, \mathbf{false}\}$. Of course, π also induces an interpretation over the points in $\mathcal{PT}(\mathcal{R})$: simply take $\pi(r, m)$ to be $\pi(r(m))$. We now define what it means for a formula ϕ to be true at a point (r, m) in an interpreted system $\mathcal{I}$, written $(\mathcal{I}, r, m) \models \phi$, by induction on the structure of formulas:

- $(\mathcal{I}, r, m) \models p$ iff $(\pi(r, m))(p) = \mathbf{true}$;
- $(\mathcal{I}, r, m) \models \phi \wedge \psi$ iff $(\mathcal{I}, r, m) \models \phi$ and $(\mathcal{I}, r, m) \models \psi$;
- $(\mathcal{I}, r, m) \models \neg\phi$ iff $(\mathcal{I}, r, m) \not\models \phi$;
- $(\mathcal{I}, r, m) \models K_i\phi$ iff $(\mathcal{I}, r', m') \models \phi$ for all $(r', m') \in \mathcal{K}_i(r, m)$;
- $(\mathcal{I}, r, m) \models \Diamond\phi$ iff there exists n such that $(\mathcal{I}, r, n) \models \phi$.

As usual, we say that ϕ is *valid in* $\mathcal{I}$ and write $\mathcal{I} \models \phi$ if $(\mathcal{I}, r, m) \models \phi$ for all points (r, m) in $\mathcal{I}$; similarly, ϕ is *satisfiable in* $\mathcal{I}$ if $(\mathcal{I}, r, m) \models \phi$ for some point (r, m)

in $\mathcal{I}$. We abbreviate $\neg K_i \neg \phi$ as $P_i \phi$. We read $P_i \phi$ as "(according to agent i) ϕ is possible." Note that $(\mathcal{I}, r, m) \models P_i \phi$ if there exists a point $(r', m') \in \mathcal{K}_i(r, m)$ such that $(\mathcal{I}, r', m') \models \phi$.

The systems framework lets us express in a natural way some standard assumptions about systems. For example, we can reason about *synchronous* systems, where agents always know the time. Formally, $\mathcal{R}$ is synchronous if, for all agents i and points (r, m) and (r', m'), if $r_i(m) = r'_i(m')$, then $m = m'$.

Another standard assumption is that agents have *perfect recall*. This assumption is implicitly made in almost all system models considered in the security literature, but is *not* implicit in the runs-and-systems model. Roughly speaking, an agent with perfect recall can reconstruct his complete local history. In synchronous systems, for example, an agent's local state changes with every tick of the external clock, so agent i's having perfect recall implies that the sequence $\langle r_i(0), \ldots, r_i(m) \rangle$ must be encoded in $r_i(m+1)$. To formalize perfect recall, let *agent i's local-state sequence at the point* (r, m) be the sequence of local states she has gone through in run r up to time m, without consecutive repetitions. Thus, if from time 0 through time 4 in run r agent i has gone through the sequence $\langle s_i, s_i, s'_i, s_i, s_i \rangle$ of local states, where $s_i \neq s'_i$, then her local-state sequence at $(r, 4)$ is $\langle s_i, s'_i, s_i \rangle$. Intuitively, an agent has perfect recall if her current local state encodes her local-state sequence. More formally, we say that *agent i has perfect recall in system $\mathcal{R}$* if, at all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, if $(r', m') \in \mathcal{K}_i(r, m)$, then agent i has the same local-state sequence at both (r, m) and (r', m'). Thus, agent i has perfect recall if she "remembers" her local-state sequence at all times. It is easy to check that perfect recall has the following key property: if $(r', m'_1) \in \mathcal{K}_i(r, m_1)$, then for all $m_2 \leq m_1$, there exists $m'_2 \leq m'_1$ such that $(r', m'_2) \in \mathcal{K}_i(r, m_2)$. (See [20] for more

discussion of this definition.)

2.2 Probabilistic multiagent systems

Many real-world systems involve nondeterminism and randomization. Adding probability to the runs-and-systems framework gives us more power to reason about such systems. We present two ways of doing so: putting probabilities on points and putting probabilities on runs.

We first consider putting probabilities on points, using a general approach described by Halpern [36]. Intuitively, at any point in time, agents have probabilities on the current state of the system (including the states of other agents). Accordingly, this approach specifies a probability on points for every agent and every point. Define a *probability assignment* $\mathcal{PR}$ to be a function that assigns to each agent i and point (r, m) a probability space

$$\mathcal{PR}(r, m, i) = (W_{r,m,i}, \mathcal{F}_{r,m,i}, \mu_{r,m,i}),$$

where $W_{r,m,i} \subseteq \mathcal{PT}(\mathcal{R})$ is i's sample space at (r, m) and $\mu_{r,m,i}$ is a probability measure defined on the subsets of $W_{r,m,i}$ in $\mathcal{F}_{r,m,i}$. (That is, $\mathcal{F}_{r,m,i}$ is a σ-algebra that defines the measurable subsets of $W_{r,m,i}$.) We call a pair $(\mathcal{R}, \mathcal{PR})$ a *probability system*.

To put probabilities on runs, we begin by assuming that there is some way to assign a probability measure to the space of runs. (Intuitively, this measure represents the objective probability with which various execution sequences will occur.) Define a *run-based probability system* to be a triple $(\mathcal{R}, \mathcal{F}, \mu)$, where $\mathcal{R}$ is a system, $\mathcal{F}$ is a σ-algebra of subsets of $\mathcal{R}$, and μ is a probability measure defined on $\mathcal{F}$. Note that a run-based probability system requires only one probability

measure, rather than a probability measure at each point and for each agent. In practice, such a measure is often relatively easy to come by. In the same way that a set of runs can be generated by a protocol, a runs-based probability system can be generated by a probabilistic protocol: the probability of a set of runs sharing a common prefix can be derived by multiplying the probabilities of the protocol transitions necessary to generate the prefix (see [36, 44] for further discussion).

Here and throughout the paper, we assume for simplicity that in a run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$, $\mathcal{F}$ contains all sets of the form $\mathcal{R}(\mathcal{K}_i(r,m))$, for all points (r,m) and all agents i. That is, if a set of runs is generated by an agent's local state, it is measurable. We also assume that $\mu(\mathcal{R}(\mathcal{K}_i(r,m))) > 0$, so that we can condition on information sets.

We now discuss an approach due to Halpern and Tuttle [44] that is useful for connection probability systems and run-based probability systems. Given an agent i and a point (r,m), we would like to derive the probability measure $\mu_{r,m,i}$ from μ by conditioning μ on $\mathcal{K}_i(r,m)$, the information that i has at the point (r,m). The problem is that $\mathcal{K}_i(r,m)$ is a set of *points*, not a set of *runs*, so straightforward conditioning does not work. To solve this problem, we condition μ on the set of runs going through $\mathcal{K}_i(r,m)$, rather than on $\mathcal{K}_i(r,m)$. Given a set U of points, let $\mathcal{R}(U)$ be the set of runs in $\mathcal{R}$ going through the points in U:

$$\mathcal{R}(U) \triangleq \{r \in \mathcal{R} : (r,m) \in U \text{ for some } m\}.$$

Conditioning is always well-defined, given our assumption that $\mathcal{R}(\mathcal{K}_i(r,m))$ has positive measure.

We can now define a measure $\mu_{r,m,i}$ on the points in $\mathcal{K}_i(r,m)$ as follows. If $\mathcal{S} \subseteq \mathcal{R}$ and $A \subseteq \mathcal{PT}(\mathcal{R})$, let $A(\mathcal{S})$ be the set of points in A that lie on runs in $\mathcal{S}$;

that is,

$$A(\mathcal{S}) \triangleq \{(r', m') \in A : r' \in \mathcal{S}\}.$$

In particular, $\mathcal{K}_i(r, m)(\mathcal{S})$ consists of the points in $\mathcal{K}_i(r, m)$ that lie on runs in $\mathcal{S}$. Let $\mathcal{F}_{r,m,i}$ consist of all sets of the form $\mathcal{K}_i(r, m)(\mathcal{S})$, where $\mathcal{S} \in \mathcal{F}$. Then define

$$\mu_{r,m,i}(\mathcal{K}_i(r, m)(\mathcal{S})) \triangleq \mu(\mathcal{S} \mid \mathcal{R}(\mathcal{K}_i(r, m)).$$

It is easy to check that if $U \subseteq \mathcal{K}_i(r, m)$ is measurable with respect with respect to $\mu_{r,m,i}$, then $\mu_{r,m,i}(U) = \mu(\mathcal{R}(U) \mid \mathcal{R}(\mathcal{K}_i(r, m)))$. We say that the resulting probability system $(\mathcal{R}, \mathcal{PR})$ is *determined* by the run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$, and call μ the *underlying measure*. We call a probability system *standard* if it is determined by a run-based probability system.

We can also extend the logic of knowledge described in the last section to probabilistic systems. Define an *interpreted probability system* $\mathcal{I}$ to be a tuple $(\mathcal{R}, \mathcal{PR}, \pi)$, where $(\mathcal{R}, \mathcal{PR})$ is a probability system. In an interpreted probability system we can give semantics to syntactic statements of probability. We are most interested in formulas of the form $\Pr_i(\phi) = \alpha$ (or similar formulas with $\leq$, $>$, etc., instead of $=$). Such formulas were given semantics by Fagin, Halpern, and Megiddo [19]; we follow their approach here. Intuitively, a formula such as $\Pr_i(\phi) = \alpha$ is true at a point (r, m) if, according to $\mu_{r,m,i}$, the probability that ϕ is true is given by α. More formally, $(\mathcal{I}, r, m) \models \Pr_i(\phi) = \alpha$ if

$$\mu_{r,m,i}(\{(r', m') \in \mathcal{K}_i(r, m) : (\mathcal{I}, r', m') \models \phi\}) = \alpha.$$

Similarly, we can give semantics to $\Pr_i(\phi) \leq \alpha$ and $\Pr(\phi) > \alpha$, etc., as well as conditional formulas such as $\Pr(\phi \mid \psi) = \alpha$. Note that although these formulas talk about probability, they are either true or false at a given state.

The semantics for a formula such as $\Pr_i(\phi)$ implicitly assumes that the set of points in $\mathcal{K}_i(r, m)$ where ϕ is true is measurable. While there are ways of dealing with non-measurable sets (see [19]), here we assume that all relevant sets are measurable. This is certainly true in synchronous standard systems determined by a a run-based system where all sets of runs are measurable. More generally, it is true in a probability system $(\mathcal{R}, \mathcal{PR})$ where, for all r, m, i, all the sets in the probability space $\mathcal{PR}(r, m, i)$ are measurable.

Chapter 3

Defining Secrecy

3.1 Secrecy in nonprobabilistic systems

3.1.1 Defining secrecy

In this section we give abstract definitions of secrecy for systems described using the runs-and-systems model. Recall that we use the term "secrecy" in a fairly specific way: roughly speaking, we define secrecy so as to ensure that low-confidentiality agents do not know anything about the state of high-confidentiality agents. (Hereafter we use the more concise terms "high" and "low" when describing the confidentiality level associated with inputs, users, and so on.) In Section 3.1.4, we formalize these intuitions using the epistemic logic of Section 2.1.

To motivate our definitions, we use programs expressed in a simple imperative programming language. (We define such a language formally in Chapter 6, Section 6.2.) We assume that these programs are executed sequentially on a single machine, and that users with different security clearances can interact with the machine via channels appropriate to their security level. For example, the command **input** x **from** H prompts the high channel H for an input and stores the input value in program variable x, while the command **output** e **to** L outputs the value of the expression e on the low channel L.

All the programs that we consider determine systems in an obvious way, once we decide whether to model the systems synchronously or asynchronously. For

example, in a synchronous system determined by the following program:

output 0 **to** L;
output 1 **to** L

the system consists of exactly one run.[1] In this run, L's local state is initially empty (i.e., $\langle\,\rangle$) and is then updated with a new output event—where $out(L, i)$ denotes the output of value i on channel L—at each time step. At time 1, L's local state is $\langle out(L, 1)\rangle$, and at time 2 it is $\langle out(L, 1), out(L, 2)\rangle$. Since there is no output event at subsequent steps, at time 4, L's local state is

$$\langle out(L, 1), out(L, 2), \_, \_\rangle.$$

L's local state is different at time 2 and time 4, since L is aware that time has passed. By way of contrast, one way to translate programs to asynchronous systems is to model agents' local states so that they are modified only when input and output event occurs on channels to which they have access. In an asynchronous system determined by the program above, L's state would be unchanged after time 2.

The strongest notion of secrecy that we consider in this section is the requirement that an agent, based on her local state, must not be able to infer anything about the local state of another agent. To guarantee that an agent i is unable to rule out any possible local states for another agent j, we require that every possible local state for j be compatible with every possible local state for i:

Definition 1 *Agent j maintains* total secrecy *with respect to i in system $\mathcal{R}$ if, for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, $\mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m') \neq \emptyset$.*

[1]Though for simplicity we ignore timing variations arising due to blocking input commands, our system model can easily handle such timing issues.

Total secrecy is a strong property. For almost any imaginable system, it is, in fact, too strong to be useful. There are two important respects in which it is too strong. First, total secrecy is not at all selective about which parts of the high agent are protected. Second, total secrecy is unreasonably demanding with respect to issues of time and synchrony. Both of these issues can be handled with appropriate weakenings of total secrecy, which are now discussed.

3.1.2 Weakening total secrecy using information functions

The first respect in which total secrecy is too strong has to do with the fact that total secrecy protects *everything* about the state of the high agent. In some systems, we might want only some part of the high agent's state to be kept secret from the low agent. For example, we might want the high agent to be able to observe details about the state of the low agent, in which case our definitions are too strong because they rule out any correlation between the states of the high and low agents.

To make this more concrete, consider the following program:

input x **from** L;
output x **to** H;
output 1 **to** L

H does not maintain total secrecy with respect to L because after L sees $out(L, 1)$ he knows that H has already seen L's first input value as her output. (Note that in a synchronous setting the final output is irrelevant: L would know that H had seen L's input value at the second time step.) If we want to protect only the input values provided by H, total secrecy is too strong. We may be interested in a weaker notion of secrecy, which allows L to realize that H knows L's input value but still keeps all of the "significant" part of H's state secret. Rather than trying

to define "significant," we characterize significance abstractly using what we call an "information function."

Definition 2 *A* j-information function *on* $\mathcal{R}$ *is a function* f *from* $\mathcal{PT}(\mathcal{R})$ *to some range that depends only on* j*'s local state; that is* $f(r,m) = f(r',m')$ *if* $r_j(m) = r'_j(m')$.

Thus, for example, if j's local state at any point (r,m) includes a list of input and output operations, $f(r,m)$ could consist of only the output values contained in j's local state. Intuitively, $f(r,m)$ is intended to represent that part of j's local state that is significant to whomever is doing the reasoning.

Definition 3 *If* f *is a* j*-information function, agent* j *maintains* total f-secrecy *with respect to* i *in system* $\mathcal{R}$ *if, for all points* (r,m) *and values* v *in the range of* f, $\mathcal{K}_i(r,m) \cap f^{-1}(v) \neq \emptyset$ *(where* $f^{-1}(v)$ *is simply the preimage of* v*, that is, all points* (r,m) *such that* $f(r,m) = v$*).*

Of course, if $f(r,m) = r_j(m)$, then $f^{-1}(r'_j(m')) = \mathcal{K}_j(r',m')$, so total secrecy is a special case of total f-secrecy.

To see how f-secrecy handles the example program above, suppose that we introduce an information function f that extracts only the input events from H's state. Because $f(r,m)$ is always empty, it is easy to see that H maintains total f-secrecy with respect to L. If our goal is to protect only the input values provided by H, any program that never reads input values from H is trivially secure.

Total f-secrecy is a special case of *nondeducibility*, introduced by Sutherland [87]. Sutherland considers "abstract" systems that are characterized by a set W of worlds. He focuses on two agents, whose views are represented by information functions g and h on W. Sutherland says that *no information flows from*

g to h if, for all worlds $w, w' \in W$, there exists some world $w'' \in W$ such that $g(w'') = g(w)$ and $h(w'') = h(w')$. This notion is often called *nondeducibility (with respect to g and h)* in the literature. To see how total f-secrecy is a special case of nondeducibility, let $W = \mathcal{PT}(\mathcal{R})$, the set of all points of the system. Given a point (r, m), let $g(r, m) = r_i(m)$. Then total f-secrecy is equivalent to nondeducibility with respect to g and f.

Note that nondeducibility is symmetric: no information flows from g to h iff no information flows from h to g. Since most standard noninterference properties focus only on protecting the state of some high agent, symmetry appears to suggest that if the actions of a high agent are kept secret from a low agent, then the actions of a low agent must also be kept secret from the high agent. Our definitions help to clarify this issue. Total secrecy as we have defined it is indeed symmetric: j maintains total secrecy with respect to i iff i maintains total secrecy with respect to j. However, total f-secrecy is not symmetric in general. If j maintains total f-secrecy with respect to i, it may not even make sense to talk about i maintaining total f-secrecy with respect to j, because f may not be an i-information function. Thus, although f-secrecy is an instantiation of nondeducibility (with respect to an appropriate g and h), the symmetry at the level of g and h does not translate to symmetry at the level of f-secrecy, which is where it matters.

While f-secrecy is useful conceptually, it is essentially a trivial technical generalization of the basic notion of secrecy, because for any agent j and j-information function f, we can reason about a new agent j_f whose local state at any point (r, m) is $r_{j_f}(m) = f(r_j, m)$. Therefore, every theorem we prove involving secrecy holds for f-secrecy as well. For this reason, and to simplify the definitions given in the remainder of the paper, we ignore information functions and deal only with se-

crecy of one agent with respect to another. Indeed, all our definitions hold without change for any agent "created" by identifying an agent with a function on global states.

3.1.3 Run-based secrecy and synchronous secrecy

The second respect in which total secrecy is too strong involves *time*. To understand the issue, consider synchronous systems (as defined in Section 2.1). In such systems, the low agent knows the time and knows that the high agent knows it too. Thus, the low agent can rule out all high states except those that occur at the current time. Even in semisynchronous systems, where agents know the time to within some tolerance ϵ, total secrecy is impossible, because low agents can rule out high states that occur only in the distant past or future.

Total secrecy may be an unreasonable condition even in asynchronous systems. To see this, consider the following program:

input x **from** H;
output 1 **to** L

Even though L does not know which value was entered by H, H does not maintain total secrecy with respect to L in this program simply because L knows, after seeing his output value, that H has already entered *some* input value. Indeed, total secrecy—and also total f-secrecy, for an information function f that extracts high input values—rules out any program where low output events follow high input events.

We now consider two distinct ways of resolving this problem. The first way weakens total secrecy by considering runs instead of points. Total secrecy (of j with respect to i) says that at all times, agent i must consider all states of j to

be (currently) possible. A weaker version of total secrecy says that at all times, i must consider it possible that every possible state of j either occurs at that time, or at some point in the past or future. We formalize this in the following definition. Given a set U of points, recall that $\mathcal{R}(U)$ is the runs in $\mathcal{R}$ going through a point in U.

Definition 4 *Agent j maintains* run-based secrecy *with respect to j in system $\mathcal{R}$ if, for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, $\mathcal{R}(\mathcal{K}_i(r, m)) \cap \mathcal{R}(\mathcal{K}_j(r', m')) \neq \emptyset$.*

It is easy to check that j maintains run-based secrecy with respect to j in system $\mathcal{R}$ iff for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, there exists a run r'' and times n and n' such that $r''_i(n) = r_i(m)$ and $r''_j(n') = r'_j(m')$. To relate the formal definition to its informal motivation, note that every state of j that occurs in the system has the form $r'_j(m')$ for some point (r', m'). Suppose that i's state is $r_i(m)$. If there exists a point (r'', n'') such that $r''_i(n'') = r_i(m)$ and $r''_j(n'') = r'_j(m')$, agent i considers it possible that j currently has state $r'_j(m')$. If instead $r''_j(n) = r'_j(m')$ for $n \langle n''$, then i currently considers it possible that j was in state $r'_j(m')$ at some point in the past; similarly, if $n > n''$, then i thinks that j could be in state $r'_j(m')$ at some point in the future. Note that total secrecy implies run-based secrecy, but the converse is not necessarily true (as shown in Example 2, in Section A.1 in the appendix). While run-based secrecy is still a very strong security property, it seems much more reasonable than total secrecy. In particular, H maintains run-based secrecy with respect to L in the system corresponding to the program **input** x **from** H; **output** 1 **to** L—as far as L is concerned, all the runs in this system look identical. However, run-based secrecy does not hold in systems derived from the kinds of programs typically used to demonstrate indirect information flows,

such as

input x **from** H;
if $(x = 0)$ **then**
  output 0 **to** L
else
  output 1 **to** L;

where run-based secrecy does not hold because L's output gives information about whether H's input value was equal to 0.

The second way to weaken total secrecy is to relax the requirement that the low agent cannot rule out any possible high states. We make this formal as follows.

Definition 5 *An* i-allowability function *on* $\mathcal{R}$ *is a function* C *from* $\mathcal{PT}(\mathcal{R})$ *to subsets of* $\mathcal{PT}(\mathcal{R})$ *such that* $\mathcal{K}_i(r,m) \subseteq C(r,m)$ *for all* $(r,m) \in \mathcal{PT}(\mathcal{R})$.

Intuitively, $\mathcal{PT}(\mathcal{R}) - C(r,m)$ is the set of points that i is allowed to "rule out" at the point (r,m). It seems reasonable to insist that the points that i considers possible at (r,m) not be ruled out, which is why we require that $\mathcal{K}_i(r,m) \subseteq C(r,m)$.

Definition 6 *If* C *is an* i*-allowability function, then* j maintains C-secrecy with respect to i *if, for all points* $(r,m) \in \mathcal{PT}(\mathcal{R})$ *and* $(r',m') \in C(r,m)$*, we have* $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m') \neq \emptyset$.

If $C(r,m) = \mathcal{PT}(\mathcal{R})$ for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$, then C-secrecy reduces to total secrecy. Synchrony can be captured by the allowability function $S(r,m) = \{(r',m) : r' \in \mathcal{R}\}$. Informally, this says that agent i is allowed to know what time it is. We sometimes call S-secrecy *synchronous secrecy*. It is easy to see that H maintains synchronous secrecy with respect to L in the system generated by the program **input** x **from** H; **output** 1 **to** L.

In synchronous systems, synchronous secrecy has a simple characterization.

Proposition 1 *Agent j maintains synchronous secrecy with respect to i in a synchronous system $\mathcal{R}$ iff, for all runs $r, r' \in \mathcal{R}$ and times m, we have that $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m) \neq \emptyset$.*

Proof: This follows trivially from the definitions. □

In synchronous input/output trace systems, synchronous secrecy is essentially equivalent to the standard notion of *separability* [58]. (Total secrecy can be viewed as an asynchronous version of separability. See Section 5.1.1 for further discussion of this issue.) The security literature has typically focused on either synchronous systems or completely asynchronous systems. One advantage of our framework is that we can easily model both of these extreme cases, as well as being able to handle in-between cases, which do not seem to have been considered up to now. Consider a semisynchronous system where agents know the time to within a tolerance of ϵ. At time 5, for example, an agent knows that the true time is in the interval $[5-\epsilon, 5+\epsilon]$. This corresponds to the allowability function $SS(r,m) = \{(r',m') : |m-m'| \leq \epsilon\}$, for the appropriate ϵ. We believe that any attempt to define security for semisynchronous systems will require something like allowability functions.

C-secrecy and run-based secrecy represent two quite different approaches to weakening total secrecy: allowability functions restrict the set of j-information sets that i must consider possible, while run-based secrecy focuses on runs rather than points. Even if we focus on synchronous secrecy, the two notions are distinct. In systems without perfect recall, for example, we may have synchronous secrecy without having run-based secrecy, while in asynchronous systems we may have run-based secrecy without having synchronous secrecy. (See Section A.1 in the appendix for examples.) However, there are contexts in which the definitions do

coincide, suggesting that they are capturing some of the same intuitions. Consider, for example, our definition of synchronous secrecy. Intuitively it might at first seem that synchronous secrecy goes too far in weakening total secrecy. Informally, j maintains *total* secrecy with respect to i if i never learns anything not only about j's current state, but also his possible future and future states. Synchronous secrecy seems only to say that i never learns anything about j's state *at the current time*. However, when agents have perfect recall, it turns out that synchronous secrecy implies run-based secrecy, thus addressing this concern.

To make this precise for a more general class of allowability functions, we need the following definition, which captures the intuition that an allowability function depends only on timing. Given any two runs, we want the allowability function to map points on the first run to contiguous, nonempty sets of points on the second run in a way that respects the ordering of points on the first run and covers all points on the second run.

Definition 7 *An allowability function* C depends only on timing *if it satisfies the following three conditions: (a) for all runs* $r, r' \in \mathcal{R}$*, and all times* m'*, there exists* m *such that* $(r', m') \in C(r, m)$*; (b) if* $(r', m') \in C(r, m)$*, and* $n \geq m$ *(resp.* $n \leq m$*), there exists* $n' \geq m'$ *(resp.* $n' \leq m'$*) such that* $(r', n') \in C(r, n)$*; (c) if* $(r', n_1) \in C(r, m)$*,* $(r', n_2) \in C(r, m)$*, and* $n_1 \leq m' \leq n_2$*, then* $(r', m') \in C(r, m)$*.*

It is easy to check that both synchronous and semi-synchronous allowability functions depend only on timing. We now show that C-secrecy implies run-based secrecy if C depends only on timing.

Proposition 2 *If* $\mathcal{R}$ *is a system where* i *and* j *have perfect recall,* C *depends only on timing, and* j *maintains* C*-secrecy with respect to* i*, then* j *maintains run-based*

secrecy with respect to i.

In synchronous systems with perfect recall, synchronous secrecy and run-based secrecy agree. This reinforces our claim that both definitions are natural, useful weakenings of total secrecy.

Proposition 3 *If $\mathcal{R}$ is a synchronous system where both i and j have perfect recall, then agent j maintains synchronous secrecy with respect to i iff j maintains run-based secrecy with respect to i.*

The requirement in Proposition 3 that both agents have perfect recall is necessary; see Example 1, in the appendix, for details. Without perfect recall, two things can go wrong. First, if i does not have perfect recall, she might be able to determine at time n what j's state is going to be at some future time $n' > n$, but then forget about it by time n', so that j maintains synchronous secrecy with respect to i, but not run-based secrecy. Second, if j does not have perfect recall, i might learn something about j's state in the past, but j might still maintain synchronous secrecy with respect to i because j has forgotten this information by the time i learns it. These examples suggest that secrecy is less interesting when agents can forget facts that they once knew. At any rate, it makes sense to model agents as if they have perfect recall, since not doing so requires us to trust that agents will forget facts when we need them to, leading to weaker security guarantees.

3.1.4 Syntactic characterizations of secrecy

Our definitions of secrecy are semantic; they are given in terms of the local states of agents. As we shall see, it is helpful to reason syntactically about secrecy, using the logic of knowledge discussed in Section 2.1. Our goal in this section is

to characterize secrecy in terms of the knowledge—or more precisely, the lack of knowledge—of the agent with respect to whom secrecy is maintained. To this end, we show that the state of an agent j is kept secret from an agent i exactly if i does not know any formulas that depend only on the state of j, or, dually, if i always thinks that any formula that depends on the state of j is currently possible.

For this characterization, we use the modal logic of knowledge described in Section 2.1. But first, we need to define what it means for a formula to depend on the local state of a particular agent. Given an agent j, a formula ϕ is *j-local* in an interpreted system $\mathcal{I}$ if, for all points (r, m) and (r', m') such that $r_j(m) = r'_j(m')$, $(\mathcal{I}, r, m) \models \phi$ iff $(\mathcal{I}, r', m') \models \phi$. It is easy to check that ϕ is j-local in $\mathcal{I}$ iff $\mathcal{I} \models K_j\phi \vee K_j\neg\phi$; thus, j-locality can be characterized syntactically. (See [17] for an introduction to the logic of local propositions.) The notion of j-locality has another useful semantic characterization:

Proposition 4 *A formula ϕ is j-local in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ iff there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \phi$ exactly when $(r, m) \in \bigcup_{\mathcal{K}\in\Omega} \mathcal{K}$.*

The following theorem shows that the semantic characterizations of secrecy given in Section 3.1.1 correspond closely to our intuitions of what secrecy should mean: agent j maintains secrecy with respect to i precisely if i cannot rule out any satisfiable facts that depend only on the local state of j.

Theorem 1 *Suppose that C is an i-allowability function. Agent j maintains C-secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π and point (r, m), if ϕ is j-local and $(\mathcal{I}, r', m') \models \phi$ for some $(r', m') \in C(r, m)$, then $(\mathcal{I}, r, m) \models P_i\phi$.*

Since total secrecy is just C-secrecy for the allowability function C such that $C(r, m)$ consists of all point in $\mathcal{R}$, the following corollary, which gives an elegant syntactic characterization of total secrecy, is immediate.

Corollary 1 *Agent j maintains total secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π, if the formula ϕ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i\phi$.*

Corollary 1 says that total secrecy requires i not to know any j-local fact that is not valid in $\mathcal{I}$. A similar result holds for synchronous secrecy. For brevity, and because we prove more general results in later sections, we ignore the details here.

We can also give a similar syntactic characterization of run-based secrecy. For j to maintain total secrecy with respect to i, if ϕ is j-local, it is always necessary for i to think that ϕ was possible. For run-based secrecy, we require only that i think that ϕ is possible sometime in the current run. Recall that the formula $\Diamond\phi$ means "ϕ is true at some point in the current run."

Theorem 2 *Agent j maintains run-based secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π, if ϕ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i\Diamond\phi$.*

The results of this section show that secrecy has a syntactic characterization that is equivalent to the semantic characterization. We speculate that this characterization may be relevant to natural generalizations of secrecy involving declassification and resource-bounded agents. With *declassification* [95, 65], system administrators may deem that some facts about high users may be leaked to low users. If a secrecy policy corresponds to a set of formulas that must be kept secret from the low agent, it seems likely that declassification could be modeled by

specifying particular formulas to be removed from the set. Another generalization of secrecy involves replacing knowledge by a more computational notion such as *algorithmic knowledge* [20, 42]. Recall that the definition of knowledge described in Section 2.1 suffers from the *logical omniscience* problem: agents know all tautologies and know all logical consequences of their knowledge [20]. In the context of security, it may not matter that an agent with unbounded computational resources can factor and decrypt a message, so long as resource-bounded agents cannot decrypt the message. It would be interesting to define secrecy with respect to resource-bounded agents by requiring only that agents do not *algorithmically know* various facts.

3.2 Secrecy in probabilistic systems

The definitions of secrecy that we have considered up to now are *possibilistic*; they consider only whether or not an event is possible. They thus cannot capture what seem like rather serious leakages of information. As a motivating example, consider a system $\mathcal{R}$ with two agents Alice and Bob, who we think of as sitting at separate computer terminals. Suppose that $\mathcal{L}$ is a language with n words. At time 1, Bob inputs a string $x \in \mathcal{L}$ chosen uniformly at random. At time 2, with probability $1-\epsilon$, the system outputs x directly to Alice's terminal. However, with probability ϵ, the system is struck by a cosmic ray as Bob's input is transmitted to Alice, and in this case the system outputs a random string from $\mathcal{L}$. (Bob receives no information about what Alice sees.) Thus, there are $n(n+1)$ possible runs in this system: n runs where no cosmic ray hits, and n^2 runs where the cosmic ray hits. Moreover, it is immediate that Bob maintains (possibilistic) synchronous secrecy with respect to Alice even though, with very high probability, Alice sees exactly

what Bob's input was.

Although this example may seem contrived, it is easy to implement in a programming language that includes operators for randomization. For example, suppose that we extend the input/output language from the last section to include an infix operator $_p[]$, where the command $c_0\ {}_p[]\ c_1$ executes c_0 with probability p and c_1 with probability $1-p$, and an operator **rand** that returns one of the n words in the language $\mathcal{L}$ with uniform probability. The following program implements the cosmic-ray system:

input w **from** B;
$\{$**output rand**() **to** $A\ {}_\epsilon[]$ **output** w **to** $A\}$

To reason about the unwanted information flow in this example, we need to add probability to the framework. We can do that by putting an obvious probability measure on the runs in $\mathcal{R}$:

- for each $x \in \mathcal{L}$, the run where Bob inputs x and no cosmic ray hits (so that Alice sees x) gets probability $(1-\epsilon)/n$.
- for each pair $(x, y) \in \mathcal{L} \times \mathcal{L}$, the run where the cosmic ray hits, Bob inputs x, and Alice sees y gets probability ϵ/n^2.

If Alice sees x, her posterior probability that Bob's input was x is

$$\Pr_{Alice}(\text{Bob typed } x \,|\, \text{Alice sees } x) = \frac{\epsilon + n - n\epsilon}{n} = 1 - \frac{n-1}{n}\epsilon.$$

If Alice sees x, her posterior probability that Bob's input was $y \neq x$ is

$$\Pr_{Alice}(\text{Bob typed } x \,|\, \text{Alice sees } y) = \frac{\epsilon}{n}.$$

Thus, if $\epsilon > 0$, even though Alice never learns *with certainty* that Bob's input was x, her probability that it was x rises from $1/n$ to almost 1 as soon as she sees an x.

In this section we introduce definitions of probabilistic secrecy. The definitions and the technical results we obtain closely resemble the definitions and results of the previous two sections. This is no coincidence. As we show in Section 3.3, probabilistic and possibilistic secrecy are instances of a definition of *plausibilistic* secrecy for which similar results can be proved in more generality.

3.2.1 Defining probabilistic secrecy

To reason about probabilistic secrecy we employ the probabilistic systems defined in Section 2.2. We can give definitions of secrecy both for probability systems (with probabilities on points) and for run-based probability systems (with probabilities on runs).

Given a probability system, we can give relatively straightforward definitions of probabilistic total secrecy and synchronous secrecy. Rather than requiring that an agent i think that all states of another j are *possible*, we require that all of those states be measurable and equally likely according to i's subjective probability measure.

Definition 8 *Agent j maintains* probabilistic total secrecy *with respect to agent i in $(\mathcal{R}, \mathcal{PR})$ if, for all points (r,m), (r',m'), and (r'',m'') in $\mathcal{PT}(\mathcal{R})$, we have $\mathcal{K}_j(r'',m'') \cap \mathcal{K}_i(r,m) \in \mathcal{F}_{r,m,i}$, $\mathcal{K}_j(r'',m'') \cap \mathcal{K}_i(r',m') \in \mathcal{F}_{r',m',i}$, and*

$$\mu_{r,m,i}(\mathcal{K}_j(r'',m'') \cap \mathcal{K}_i(r,m)) = \mu_{r',m',i}(\mathcal{K}_j(r'',m'') \cap \mathcal{K}_i(r',m')).$$

Probabilistic total secrecy is a straightforward extension of total secrecy. Indeed, if for all points (r,m) we have $\mu_{r,m,i}(\{(r,m)\}) > 0$, then probabilistic total secrecy implies total secrecy (as in Definition 1).

Proposition 5 *If $(\mathcal{R}, \mathcal{PR})$ is a probability system such that $\mu_{r,m,i}(\{(r,m)\}) > 0$ for all points (r,m) and j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, then j also maintains total secrecy with respect to i in $\mathcal{R}$.*

Like total secrecy, probabilistic total secrecy is an unrealistic requirement in practice, and cannot be achieved in synchronous systems. To make matters worse, the sets $\mathcal{K}_j(r'', m'') \cap \mathcal{K}_i(r,m)$ are typically not measurable according to what is perhaps the most common approach for defining $\mathcal{PR}$, as we show in the next section. Thus, even in completely asynchronous systems, total secrecy is usually impossible to achieve for measurability reasons alone. Fortunately, the obvious probabilistic analogue of synchronous secrecy is a more reasonable condition.

Definition 9 *Agent j maintains* probabilistic synchronous secrecy *with respect to agent i in $(\mathcal{R}, \mathcal{PR})$ if, for all runs r, r', r'' and all times m, we have $\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r,m) \in \mathcal{F}_{r,m,i}$, $\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r',m) \in \mathcal{F}_{r',m,i}$, and*

$$\mu_{r,m,i}(\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r,m)) = \mu_{r',m,i}(\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r',m)).$$

Note that if we set up the cosmic-ray system of the previous section as a probability system in such a way that Alice's probability on points reflects the posterior probabilities we described for the system, it is immediate that Bob does *not* maintain probabilistic synchronous secrecy with respect to Alice.

We now consider definitions of probabilistic secrecy for run-based probability systems. Recall from Section 3.1.1 that run-based total secrecy requires that, for all points (r,m) and (r',m'), we have $\mathcal{R}(\mathcal{K}_i(r,m)) \cap \mathcal{R}(\mathcal{K}_j(r',m')) \neq \emptyset$. In other words, run-based total secrecy is a property based on what can happen during runs, rather than points. In a run-based probability system where all information sets have positive measure, it is easy to see that this is equivalent to the requirement

that $\mu(\mathcal{R}(\mathcal{K}_j(r', m')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) > 0$. We strengthen run-based total secrecy by requiring that these probabilities be *equal*, for all i-information sets.

Definition 10 *Agent j maintains* run-based probabilistic secrecy *with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$ if for any three points $(r, m), (r', m'), (r'', m'') \in \mathcal{PT}(\mathcal{R})$,*

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r', m'))).$$

The probabilities for the cosmic-ray system were defined on sets of runs. Moreover, facts such as "Alice sees x" and "Bob typed y" correspond to information sets, exactly as in the definition of run-based probabilistic secrecy. It is easy to check that Bob does not maintain run-based probabilistic secrecy with respect to Alice.

In Section 3.2.2, we consider the connection between probability measures on points and on runs, and the corresponding connection between probabilistic secrecy and run-based probabilistic secrecy. For the remainder of this section, we consider symmetry in the context of probabilistic secrecy. In Section 3.1.1, we mentioned that our definitions of secrecy were symmetric in terms of the agents i and j. Perhaps surprisingly, probabilistic secrecy is also a symmetric condition, at least in most cases of interest. This follows from a deeper fact: under reasonable assumptions, secrecy (of j with respect to i) implies the probabilistic independence of i-information sets and j-information sets. (See Lemma 1 in the appendix for more details.)

Consider probabilities on points: if there is no connection whatsoever between $\mathcal{PR}(r, m, i)$ and $\mathcal{PR}(r, m, j)$ in a probability system $(\mathcal{R}, \mathcal{PR})$, there is obviously no reason to expect secrecy to be symmetric. However, if we assume that the probabilities of i and j at (r, m) are derived from a single common probability

measure by conditioning, then symmetry follows. This assumption, which holds for the probability systems we will consider here (and is standard in the economics literature [64]), is formalized in the next definition.

Definition 11 *A probability system* $(\mathcal{R}, \mathcal{PR})$ *satisfies the* common prior assumption *if there exists a probability space* $(\mathcal{PT}(\mathcal{R}), \mathcal{F}_{cp}, \mu_{cp})$ *such that for all agents* i *and points* $(r, m) \in \mathcal{PT}(\mathcal{R})$*, we have* $\mathcal{K}_i(r, m) \in \mathcal{F}_W$*,* $\mu_{cp}(\mathcal{K}_i(r, m)) > 0$*, and*

$$\mathcal{PR}_i(r, m) = (\mathcal{K}_i(r, m), \{U \cap K_i(r, m) \,|\, U \in \mathcal{F}_W\}, \mu_{cp} \,|\, \mathcal{K}_i(r, m)).^2$$

In probability systems that satisfy the common prior assumption, probabilistic secrecy is symmetric.

Proposition 6 *If* $(\mathcal{R}, \mathcal{PR})$ *is a probability system (resp., synchronous probability system) that satisfies the common prior assumption with prior probability* μ_{cp}*, the following are equivalent:*

(a) Agent j *maintains probabilistic total (resp., synchronous) secrecy with respect to* i*.*

(b) Agent i *maintains probabilistic total (resp., synchronous) secrecy with respect to* j*.*

(c) For all points (r, m) *and* (r', m')*,*

$$\mu_{cp}(\mathcal{K}_j(r', m') \,|\, \mathcal{K}_i(r, m)) = \mu_{cp}(\mathcal{K}_j(r', m'))$$

(resp., for all points (r, m) *and* (r', m)*,*

$$\mu_{cp}(\mathcal{K}_j(r', m) \,|\, \mathcal{K}_i(r, m)) = \mu_{cp}(\mathcal{K}_j(r', m) \,|\, \mathcal{PT}(m)),$$

[2]Actually, it is more standard in the economics literature not to require that $\mu_{cp}(\mathcal{K}_i(r, m)) > 0$. No requirements are placed on $\mu_{r,m,i}$ if $\mu_{cp}(\mathcal{K}_i(r, m)) = 0$. See [35] for a discussion of this issue.

where $\mathcal{PT}(m)$ is the set of points occurring at time m; that is, the events $\mathcal{K}_i(r,m)$ and $\mathcal{K}_j(r',m)$ are conditionally independent with respect to μ_{cp}, given that the time is m).

In run-based probability systems there is a single measure μ that is independent of the agents, and we have symmetry provided that the system is synchronous or both agents have perfect recall. (If neither condition holds, secrecy may not be symmetric, as illustrated by Example 2 in the appendix.)

Proposition 7 *If $(\mathcal{R},\mathcal{F},\mu)$ is a run-based probability system that is either synchronous or one where agents i and j both have perfect recall, then the following are equivalent:*

(a) Agent j maintains run-based probabilistic secrecy with respect to i.

(b) Agent i maintains run-based probabilistic secrecy with respect to j.

(c) For all points $(r,m),(r',m') \in \mathcal{PT}(\mathcal{R})$, $\mathcal{R}(\mathcal{K}_i(r,m))$ and $\mathcal{R}(\mathcal{K}_j(r',m'))$ are probabilistically independent with respect to μ.

3.2.2 Secrecy in standard probability systems

In Section 2.2 we defined standard probability systems to be those that are determined by run-based probability systems using the approach of Halpern and Tuttle [44]. Note that *synchronous* standard probability systems satisfy the common prior assumption, as defined in the previous section. If we assume that all runs are measurable, then we can simply take $\mu_{cp}(r,m) = \mu(r)/2^{m+1}$. This ensures that the time m points have the same relative probability as the runs, which is exactly what is needed. More generally, if $\mathcal{PT}(m)$ is the set of time m points and $\mathcal{S}$

is a measurable subset of $\mathcal{R}$, we take $\mu_{cp}(\mathcal{PT}(m)(\mathcal{S})) = \mu(\mathcal{S})/2^{m+1}$. It follows from Proposition 6 that probabilistic synchronous secrecy is symmetric in synchronous standard systems.

In synchronous standard systems with perfect recall, probabilistic secrecy and run-based probabilistic secrecy coincide. (We remark that Example 1 shows that the requirement of perfect recall is necessary.) This provides further evidence that our notion of synchronous secrecy is appropriate in synchronous systems.

Proposition 8 *If $(\mathcal{R}, \mathcal{PR})$ is the standard system determined by the synchronous run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$, and agents i and j have perfect recall in $\mathcal{R}$, then agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$ iff j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$.*

3.2.3 Characterizing probabilistic secrecy

We now demonstrate that we can characterize probabilistic secrecy syntactically, as in the nonprobabilistic case, using the definition of an interpreted probability system given in Section 2.2.

The first result shows that we can characterize probabilistic total and synchronous secrecy.

Theorem 3 *(a) If $(\mathcal{R}, \mathcal{PR})$ is a probabilistic system, then agent j maintains probabilistic total secrecy with respect to agent i iff, for every interpretation π and formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\phi) = \sigma$.*

(b) If $(\mathcal{R}, \mathcal{PR})$ is a synchronous probabilistic system, then agent j maintains probabilistic synchronous secrecy with respect to agent i iff, for every inter-

pretation π, time m, and formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\phi) = \sigma_m$ for all runs $r \in \mathcal{R}$.

We can also characterize run-based secrecy in standard systems using the $\Diamond$ operator. For this characterization, we need the additional assumption of perfect recall.

Theorem 4 *If $(\mathcal{R}, \mathcal{PR})$ is a standard probability system where agent j has perfect recall, then agent j maintains run-based probabilistic secrecy with respect to agent i iff, for every interpretation π and every formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\Diamond\phi) = \sigma$.*

Example 3 demonstrates that the assumption of perfect recall is necessary in Theorem 4 and that synchrony alone does not suffice.

3.2.4 Secrecy in adversarial systems

It is easy to capture our motivating cosmic-ray system example using a synchronous standard system because we assumed a probability on the set of runs. Furthermore, it is not hard to show that Bob does not maintain synchronous secrecy with respect to Alice in this system. However, there is an important and arguably inappropriate assumption that was made when we modeled the cosmic-ray system, namely, that we were given the probability with which Bob inputs various strings. While we took that probability to be uniform, it was not necessary to do so: any other probability distribution would have served to make our point. The critical assumption was that there is some well-defined distribution that is known to the modeler. However, in many cases the probability distribution is *not* known. In the cosmic-ray example, if we think of the strings as words in natural language, it may not be reasonable to

view all strings as equally likely. Moreover, the probability of a string may depend on the speaker: it is unlikely that a teenager would have the same distribution as an adult, or that people having a technical discussion would have the same distribution as people discussing a movie.

There are many settings in which it makes sense to reason about the nondeterminism of a system in terms of an initial nondeterministic step followed by a sequence of deterministic or probabilistic steps. The nondeterministic step could determine the choice of speaker, the adversary's protocol, or the input to a probabilistic protocol. Indeed, it has been argued [70, 90] that any setting where there is a mix of nondeterministic, probabilistic, and deterministic moves can be reduced to one where there is an initial nondeterministic move followed by probabilistic or deterministic moves. In such a setting, we do not have one probability distribution over the runs in a system. Rather, we can partition the set of runs according to the nondeterministic initial step, and then use a separate probability distribution for the set of runs corresponding to each initial step. For example, consider a setting with a single agent and an adversary. Suppose that the agent uses a protocol p and the adversary uses one of a set $\{q_1, \ldots, q_n\}$ of protocols. The system $\mathcal{R}$ consists of all the runs generated by running (p, q_k) for $k = 1, \ldots, n$. $\mathcal{R}$ can then be partitioned into n subsets $D_1, \ldots, D_n$, where D_j consists the runs of the joint protocol (p, q_j). While we may not want to assume a probability on how likely the adversary is to use q_j, typically there is a natural probability distribution on each set D_j. Note that we can capture uncertainty about a speaker's distribution over natural language strings in the same way; each protocol corresponds to a different speaker's "string-production algorithm."

Situations where there is a nondeterministic choice followed by a sequence of

probabilistic or deterministic choices can be characterized by an *adversarial probability system*, which is a tuple $(\mathcal{R}, \mathcal{D}, \Delta)$, where $\mathcal{R}$ is a system, $\mathcal{D}$ is a countable partition of $\mathcal{R}$, and $\Delta \triangleq \{(D, \mathcal{F}_D, \mu_D) : D \in \mathcal{D}\}$ is a set of probability spaces, where μ_D is a probability measure on the σ-algebra $\mathcal{F}_D$ (on $D \in \mathcal{D}$) such that, for all agents i, points (r, m), and cells D, $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \in \mathcal{F}_D$ and, if $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$, then $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m))) > 0$.[3]

There are several ways of viewing the cosmic-ray example as an adversarial probability system. If we view the input as a nondeterministic choice, then we can take $D(x)$ to consist of all runs where the input is x, and let $\mathcal{D} \triangleq \{D(x) : x \in \mathcal{L}\}$. The measure μ_x on $D(x)$ is obvious: the one run in $D(x)$ where the cosmic ray does not strike gets probability $1 - \epsilon$; the remaining n runs each get probability ϵ/n. Note that we can assign a probability on $D(x)$ without assuming anything about Bob's input distribution. Alternatively, we can assume there are k "types" of agents (child, teenager, adult, etc.), each with their own distribution over inputs. Then the initial nondeterministic choice is the type of agent. Thus, the set of runs is partitioned into sets D_j, $j = 1, \ldots, k$. We assume that agents of type j generate inputs according to probability $\Pr_j$. In each set D_j, there is one run where Bob inputs x and the cosmic ray does not strike; it has probability $\Pr_j(x)(1-\epsilon)$. There are n runs where Bob inputs x and the cosmic ray strikes; each gets probability $\Pr_j(x)\epsilon/n$.

For another example of a system where initial nondeterministic choices play an important role, consider the following program, an insecure one-time pad imple-

[3]We actually should have written $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m)) \cap D)$ rather than $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m)))$ here, since $\mathcal{R}(\mathcal{K}_i(r, m))$ is not necessarily in $\mathcal{F}_D$ (and is certainly not in $\mathcal{F}_D$ if $\mathcal{R}(\mathcal{K}_i(r, m))$ is not a subset of D). For brevity we shall continue to abuse notation and write $\mu_D(U)$ as shorthand for $\mu_D(U \cap D)$.

mentation first described by Wittbold and Johnson [93]:

$$
\begin{array}{ll}
P_1: & \textbf{while } (\textbf{true}) \textbf{ do} \\
 & \quad x := 0 \ {}_{0.5}[] \ \ x := 1; \\
 & \quad \textbf{output } x \textbf{ to } H; \\
 & \quad \textbf{input } y \textbf{ from } H; \\
 & \quad \textbf{output } x \oplus y \textbf{ to } L
\end{array}
$$

(We assume that H can input only 0 and 1 and that $\oplus$ is the exclusive-or operator.) Note that, in every iteration of this loop, H can transmit a bit b to L by choosing $x \oplus b$ as his input value. More generally, given a bit string $z = z_1 \ldots z_k$, H can transmit z to L by giving $x_i \oplus z_i$ as input at the ith iteration (where x_i is H's output value). Thus, even though H's input values are kept completely secret from L—they are encrypted with a one-time pad that L cannot see—H can transmit arbitrary messages to L. Clearly there is a sense in which Program P_1 is completely insecure.

To model the system generated by P_1 in a way that demonstrates the lack of secrecy, we need somehow to reason about the "intention" of H. One way to do so is to assume that a string z is encoded in H's initial local state and that H follows the protocol suggested above, choosing the input value $x_i \oplus z_i$. If f_{string} is an H-information function that extracts the string from H's local state, then requiring H to maintain some form of f_{string}-secrecy with respect to L would disallow the information leak in the program above.

Although it may seem strange to be concerned about preserving the secrecy of an agent who is actively attempting to transmit secret information, this turns out to be a reasonable way to capture threats such as "rogue agents" or Trojan-horse programs whose goal is to leak confidential information to public users. Such a threat model has been the motivation for many definitions of noninterference. Intuitively, an agent j can interfere with another agent i if i's state might depend

on what j does. Though some papers have suggested otherwise [56], we claim that nondeducibility-based definitions of secrecy provide a sensible way to reason about noninterference. If i's state depends on what j does, a reasonable model of j's local state should include information about the actions she can take that affect i. When this is the case, i's local state is correlated with j's local state if j interferes with i, so j preserves secrecy with respect to i only if j does not interfere with i.

We can identify an adversarial probability system with a set of run-based probability systems, by viewing the measures in Δ as constraints on a single measure on $\mathcal{R}$. Let $\mathcal{F}_{\mathcal{D}} \triangleq \sigma(\bigcup_{D \in \mathcal{D}} \mathcal{F}_D)$, the σ-algebra generated by the measurable sets of the probability spaces of Δ. (It is straightforward to check that $U \in \mathcal{F}_{\mathcal{D}}$ iff $U = \bigcup_{D \in \mathcal{D}} U_D$, where $U_D \in \mathcal{F}_D$.) Let $\mathcal{M}(\Delta)$ consist of all measures μ on $\mathcal{F}$ such that (1) for all $D \in \mathcal{D}$, if $\mu(D) > 0$ then $\mu \,|\, D = \mu_D$ (i.e., μ conditioned on D is μ_D) and (2) for all agents i and points (r, m), there exists some cell D such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mu(D) > 0$. It follows from these requirements and our assumption that if $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ then $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m) \cap D) > 0$ that $\mu(\mathcal{R}(\mathcal{K}_i(r, m)) > 0$ for all agents i and points (r, m). We can thus associate $(\mathcal{R}, \mathcal{D}, \Delta)$ with the set of run-based probability systems $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mu)$, for $\mu \in \mathcal{M}(\Delta)$.

Rather than defining secrecy in adversarial systems directly, we give a slightly more general definition. Define a *generalized run-based probability system* to be a tuple $(\mathcal{R}, \mathcal{F}, \mathcal{M})$, where $\mathcal{M}$ is a set of probability measures on the σ-algebra $\mathcal{F}$. Similarly, define a *generalized probability system* to be a tuple $(\mathcal{R}, \mathbf{PR})$, where $\mathbf{PR}$ is a set of probability assignments. We can define secrecy in generalized (run-based) probability systems by considering secrecy with respect to each probability measure/probability assignment.

Definition 12 *Agent j maintains* probabilistic total (resp. synchronous) secrecy

with respect to agent i in the generalized probabilistic system $(\mathcal{R}, \mathbf{PR})$ if, for all $\mathcal{PR} \in \mathbf{PR}$, j maintains probabilistic total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$. Agent j maintains run-based secrecy *with respect to agent i in the generalized probabilistic run-based system $(\mathcal{R}, \mathcal{F}, \mathcal{M})$ if, for all $\mu \in \mathcal{M}$, j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$.*

It is now straightforward to define secrecy in an adversarial systems by reducing it to a generalized probabilistic system. Agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{D}, \Delta)$ if j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mathcal{M}(\Delta))$. Similarly, agent j maintains total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathcal{D}, \Delta)$ if j maintains total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathbf{PR})$, where $\mathbf{PR}$ consists of all the probability assignments determined by the run-based probability systems $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mu)$ for $\mu \in \mathcal{M}(\Delta)$. A straightforward analogue of Proposition 7 holds for adversarial systems; again, secrecy is symmetric in the presence of assumptions such as perfect recall or synchrony.

3.2.5 Secrecy and evidence

Secrecy in adversarial probability systems turns out to be closely related to the notion of *evidence* in hypothesis testing (see [50] for a good overview of the literature). Consider this simple example: someone gives you a coin, which may be fair or may be double-headed. You have no idea what the probability is that the coin is fair, and it may be exceedingly unlikely that the coin is double-headed. But suppose you then observe that the coin lands heads on each of 1,000 consecutive tosses. Clearly this observation provides strong *evidence* in favor of the coin being double-headed.

In this example there are two hypotheses: that the coin is fair and that it is double-headed. Each hypothesis places a probability on the space of observations. In particular, the probability of seeing 1000 heads if the coin is fair is $1/2^{1000}$, and the probability of seeing 1000 heads if the coin is double-headed is 1. While we can talk of an observation being more or less likely with respect to each hypothesis, making an observation does *not* tell us how likely an hypothesis is. No matter how many heads we see, we do not know the probability that the coin is double-headed unless we have the prior probability of the coin being double headed. In fact, a straightforward computation using Bayes' Rule shows that if the prior probability of the coin being double-headed is α, then the probability of the coin being double-headed after seeing 1000 heads is $\frac{\alpha}{\alpha+((1-\alpha)/2^{1000}}$.

In an adversarial probability system $(\mathcal{R}, \mathcal{D}, \Delta)$, the initial nondeterministic choice plays the role of an hypothesis. For each $D \in \mathcal{D}$, μ_D can be thought of as placing a probability on observations, given that choice D is made. These observations then give evidence about the choice made. Agent i does not obtain evidence about which choice was made if the probability of any sequence of observations is the same for all choices.

Definition 13 *Agent i* obtains no evidence for the initial choice *in the adversarial probability system $(\mathcal{R}, \mathcal{D}, \Delta)$ if, for all $D, D' \in \mathcal{D}$ and all points (r, m) such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D' \neq \emptyset$, we have*

$$\mu_D(\mathcal{R}(\mathcal{K}_i(r, m))) = \mu_{D'}(\mathcal{R}(\mathcal{K}_i(r, m))).$$

Roughly speaking, i obtains no evidence for initial choices if the initial choices (other than i's own choice) are all secret. The restriction to cells such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D' \neq \emptyset$ ensures that D and D' are both

compatible with i's initial choice.

To relate this notion to secrecy, we consider adversarial probability systems with a little more structure. Suppose that for each agent $i = 1, \ldots, n$, there is a set $INIT_i$ of possible initial choices. (For example, $INIT_i$ could consist of a set of possible protocols or a set of possible initial inputs.) Let $INIT = INIT_1 \times \cdots \times INIT_n$ consist of all tuples of initial choices. For $y_i \in INIT_i$, let D_{y_i} consist of all runs in $\mathcal{R}$ where agent i's initial choice is y_i; if $y = (y_1, \ldots, y_n) \in INIT$, then $D_y = \cap_{i=1}^n D_{y_i}$ consists of all runs where the initial choices are characterized by y. Let $\mathcal{D} = \{D_y : y \in INIT\}$. To model the fact that i is aware of his initial choice, we require that for all points (r, m) and agents i, there exists y such that $\mathcal{R}(\mathcal{K}_i(r, m)) \subseteq D_y$. If $\mathcal{D}$ has this form and each agent i is aware of his initial choice, we call $(\mathcal{R}, \mathcal{D}, \Delta)$ the adversarial system *determined by INIT*.

If i obtains no evidence for the initial choice, she cannot learn anything about the initial choices of other agents. To make this precise in our framework, let $\mathcal{M}_i^{INIT}(\Delta)$ consist of the measures $\mu \in \mathcal{M}(\Delta)$ such that for all cells $D_{(y_1,\ldots,y_n)}$, we have $\mu(D_{(y_1,\ldots,y_n)}) = \mu(D_{y_i}) \cdot \mu(\cap_{j \neq i} D_{y_j})$, that is, such that the initial choices made by agent i are independent of the choices made by other agents. Intuitively, if the choices of i and the other agents are correlated, i learns something about the other agents' choices simply by making his own choice. We want to rule out such situations. Note that because all the information sets have positive probability (with respect to all $\mu \in \mathcal{M}(\Delta)$) and, for all i, there exists an information set $\mathcal{K}_i(r, m)$ such that $D_{y_i} \supseteq \mathcal{R}(\mathcal{K}_i(r, m))$, the sets D_{y_i} must also have positive probability. It follows that $INIT$ and $\mathcal{D}$ must be countable.

Given i, let i^- denote the "group agent" consisting of all agents other than i. (In particular, if the system consists of only two agents, then i^- is the agent other

than i.) The local state of i^- is just the tuple of local states of all the agents other than i. Let f_{i^-} be the i^--information function that maps a global state to the tuple of (i^-)'s initial choice. As we observed in Section 3.1.1, our definitions apply without change to new agents that we "create" by identifying them with functions on global states. In particular, our definitions apply to i^-.

Theorem 5 *Let $(\mathcal{R}, \mathcal{D}, \Delta)$ be the adversarial probability system determined by INIT and suppose that $\mathcal{R}$ is either synchronous or a system where i has perfect recall. Agent i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$ iff agent i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$.*

The assumption of either synchrony or perfect recall is necessary because the proof relies on the symmetry of run-based secrecy (as established by Proposition 7). We do not need to assume perfect recall for agent i^- because the theorem deals with f_{i^-}-secrecy and, on every run, f_{i^-} is constant. It therefore follows that the "agent" associated with f_{i^-} (in the sense described in Section 3.1.1) has perfect recall even if i^- does not.

Thinking in terms of evidence is often simpler than thinking in terms of run-based probabilistic secrecy. Moreover, the evidence-based definition of secrecy is well-defined even when the set *INIT* of initial choices is uncountable. The connection between evidence and secrecy is particularly relevant when it comes to relating our work to that of Gray and Syverson [32]; see Section 5.1.2.

3.3 Plausibilistic secrecy

So far, we have given definitions of secrecy for nonprobabilistic systems, for probability systems (where uncertainty is represented by a single probability measure), and for generalized probability systems (where uncertainty is represented by a set of probability measures). All of these definitions turn out to be special cases of secrecy with respect to a general representation of uncertainty called a *plausibility measure* [26, 27]. By giving a general definition, we can focus on the essential features of all the definitions, as well as point the way to defining notions of secrecy with respect to other representations of uncertainty that may be useful in practice.

The definitions of plausibility measures and plausibility spaces presented here are adapted from previous work [26, 27, 34, 36]. Our primary contributions are definitions of plausibilistic secrecy, as well as generalizations of several of the results described in earlier sections of this chapter.

Recall that a probability space is a tuple $(W, \mathcal{F}, \mu)$, where W is a set of worlds, $\mathcal{F}$ is an algebra of measurable subsets of W, and μ maps sets in $\mathcal{F}$ to elements of $[0, 1]$ such that the axioms of probability are satisfied. A plausibility space is a direct generalization of a probability space. We simply replace the probability measure μ with a plausibility measure Pl, which maps from sets in $\mathcal{F}$ to elements of an arbitrary partially ordered set. If $\mathrm{Pl}(A) \leq \mathrm{Pl}(B)$, then B is at least as plausible as A. Formally, a plausibility space is a tuple $(W, \mathcal{F}, \mathbf{D}, \mathrm{Pl})$, where $\mathbf{D}$ is a domain of plausibility values partially ordered by a relation $\leq_{\mathbf{D}}$, and where Pl maps from sets in $\mathcal{F}$ to elements of $\mathbf{D}$ in such a way that if $U \subseteq V$, then $\mathrm{Pl}(U) \leq_{\mathbf{D}} \mathrm{Pl}(V)$. We assume that $\mathbf{D}$ contains elements top and bottom, denoted $\top_{\mathbf{D}}$ and $\bot_{\mathbf{D}}$, such that $\mathrm{Pl}(W) = \top_{\mathbf{D}}$ and $\mathrm{Pl}(\emptyset) = \bot_{\mathbf{D}}$.

As shown in [26, 36], all standard representations of uncertainty can be viewed

as instances of plausibility measures. We consider a few examples here that will be relevant to our discussion:

- It is straightforward to see that a probability measure μ on a space W is also a plausibility measure by taking $\mathbf{D} = [0,1]$, $\mathrm{Pl} = \mu$, and $\leq_{\mathbf{D}} = \leq$.

- We can capture the notion of "possibility" using the trivial plausibility measure $\mathrm{Pl}_{\mathrm{triv}}$ that assigns the empty set plausibility 0 and all other sets plausibility 1. That is, $\mathbf{D} = \{0,1\}$, $\leq_{\mathbf{D}} = \leq$, $\mathrm{Pl}_{\mathrm{triv}}(\emptyset) = 0$, and $\mathrm{Pl}_{\mathrm{triv}}(U) = 1$ if $U \neq \emptyset$. (Intuitively, an event is possible if there is some world in which it occurs.)

- A set $\mathcal{M}$ of probability measures on a space W can be viewed as a single plausibility measure. In the special case where $\mathcal{M}$ is a finite set, say $\mathcal{M} = \{\mu_1, \ldots, \mu_n\}$, we can take $\mathbf{D}_{\mathcal{M}}$ to consist of n-tuples in $[0,1]^n$, with the pointwise ordering, and define $\mathrm{Pl}_{\mathcal{M}}(U) \triangleq (\mu_1(U), \ldots, \mu_n(U))$. Clearly $\mathrm{Pl}_{\mathcal{M}}(\emptyset) = (0, \ldots, 0)$ and $\mathrm{Pl}_{\mathcal{M}}(W) = (1, \ldots, 1)$, so $\bot_{\mathbf{D}_{\mathcal{M}}} = (0, \ldots, 0)$ and $\top_{\mathbf{D}_{\mathcal{M}}} = (1, \ldots, 1)$. If $\mathcal{M}$ is infinite, we consider a generalization of this approach. Let $\mathbf{D}_{\mathcal{M}}$ consist of all functions from $\mathcal{M}$ to $[0,1]$. The pointwise order on functions gives a partial order on $\mathbf{D}_{\mathcal{M}}$; thus, $\bot_{\mathbf{D}_{\mathcal{M}}}$ is the constant function 0, and $\top_{\mathbf{D}_{\mathcal{M}}}$ is the constant function 1. Define the plausibility measure $\mathrm{Pl}_{\mathcal{M}}$ by taking $\mathrm{Pl}_{\mathcal{M}}(U)$ to be the function f_U such that $f_U(\mu) = \mu(U)$, for all $\mu \in \mathcal{M}$.

We can define secrecy using plausibility measures by direct analogy with the probabilistic case. Given a system $\mathcal{R}$, define a *plausibility assignment* $\mathcal{PL}$ on $\mathcal{R}$ to be a function that assigns to each agent i and point (r,m) a plausibility space $(\mathcal{W}_{r,m,i}, \mathcal{F}_{r,m,i}, \mathrm{Pl}_{r,m,i})$; define a *plausiblity system* to be a pair $(\mathcal{R}, \mathcal{PL})$, where $\mathcal{PL}$

is a plausibility assignment on $\mathcal{R}$. We obtain definitions of total plausibilistic secrecy and synchronous plausibilistic secrecy by simply replacing "probability" by "plausibility" in Definitions 8 and 9.

Given a plausibility measure Pl on a system $\mathcal{R}$, we would like to define run-based plausibilistic secrecy and repeat the Halpern-Tuttle construction to generate standard plausibilistic systems. To do this, we need a notion of conditional plausibility. To motivate the definitions to come, we start by describing conditional probability spaces. The essential idea behind conditional probability spaces, which go back to Popper [69] and de Finetti [12], is to treat conditional probability, rather than unconditional probability, as the primitive notion. A *conditional probability measure* μ takes two arguments V and U; $\mu(V, U)$ is generally written $\mu(V \mid U)$. Formally, a *conditional probability space* is a tuple $(W, \mathcal{F}, \mathcal{F}', \mu)$ such that $\mathcal{F}$ is a σ-algebra over W, $\mathcal{F}'$ is a nonempty subset of $\mathcal{F}$ that is closed under supersets in $\mathcal{F}$ (so that if $U \in \mathcal{F}'$, $U \subseteq V$, and $V \in \mathcal{F}$, then $V \in \mathcal{F}'$), the domain of μ is $\mathcal{F} \times \mathcal{F}'$, and the following conditions are satisfied:

- $\mu(U \mid U) = 1$ if $U \in \mathcal{F}'$.
- if $U \in \mathcal{F}'$ and $V_1, V_2, V_3, \ldots$ are pairwise disjoint elements of $\mathcal{F}$, then
$$\mu(\cup_{i=1}^{\infty} V_i \mid U) = \sum_{i=1}^{\infty} \mu(V_i \mid U);$$
- $\mu(U_1 \cap U_2 \mid U_3) = \mu(U_1 \mid U_2 \cap U_3) \cdot \mu(U_2 \mid U_3)$ if $U_1 \in \mathcal{F}$ and $U_2 \cap U_3 \in \mathcal{F}'$.

The first two requirements guarantee that, for each fixed $U \in \mathcal{F}'$, $\mu(\cdot \mid U)$ is an unconditional probability measure. The last requirement guarantees that the various conditional probability measures "fit together." As is standard, we identify unconditional probability with conditioning on the whole space, and write $\Pr(U)$ as an abbreviation for $\Pr(U \mid W)$.

Given an unconditional probability space $(W, \mathcal{F}, \mu)$, we immediately obtain a conditional probability space by taking $\mathcal{F}'$ to consist of all sets U such that $\mu(U) \neq 0$ and defining conditional probability in the standard way. However, starting with conditional probability is more general in the sense that it is possible to extend an unconditional probability space to a conditional probability space where $\mathcal{F}'$ contains sets U such that $\mu(U) = 0$. In other words, there exist conditional probability spaces $(W, \mathcal{F}, \mathcal{F}', \mu)$ such that $\mu(U \mid W) = 0$ for some $U \in \mathcal{F}'$. This generality is useful for reasoning about secrecy, because (as we shall see) it is sometimes useful to be able to condition on sets that have a probability of 0.

To generalize conditional probability to the plausibilistic setting, we need to define operators $\oplus$ and $\otimes$ that act as analogues of $+$ and $\times$ for probability; these operators add useful algebraic structure to the plausibility spaces we consider. We extend the notion of an *algebraic plausibility spaces* [26, 34, 36] to allow an analogue of countable additivity. We briefly sketch the relevant details here.

A *countably-additive algebraic conditional plausibility space* (cacps) is a tuple $(W, \mathcal{F}, \mathcal{F}', \mathrm{Pl})$ such that

- $\mathcal{F}$ is a σ-algebra of subsets of W;
- $\mathcal{F}'$ is a nonempty subset of $\mathcal{F}$ that is closed under supersets in $\mathcal{F}$;
- there is a partially-ordered domain $\mathbf{D}$ such that, for each $V \in \mathcal{F}'$, $\mathrm{Pl}(\cdot \mid V)$ is a plausibility measure on $(W, \mathcal{F})$ with range $\mathbf{D}$ (so, intuitively, the events in $\mathcal{F}'$ are the ones for which conditioning is defined); and
- there are functions $\oplus : \mathbf{D}^\infty \to \mathbf{D}$ and $\otimes : \mathbf{D} \times \mathbf{D} \to \mathbf{D}$ such that:
  - if $U \in \mathcal{F}'$, $V_1, V_2, \ldots$, are pairwise disjoint elements of $\mathcal{F}$, and J is some

subset of $\{1, 2, 3, \ldots\}$ such that $\mathrm{Pl}(V_i) = \bot$ exactly when $i \in J$, then

$$\mathrm{Pl}(\cup_{i=1}^{\infty}(V_i \mid U)) = \oplus_{i \notin J}\mathrm{Pl}(V_i \mid U).$$

– if $U_1, U_2, U_3 \in \mathcal{F}$ and $U_2 \cap U_3 \in \mathcal{F}'$, then

$$\mathrm{Pl}(U_1 \cap U_2 \mid U_3) = \mathrm{Pl}(U_1 \mid U_2 \cap U_3) \otimes \mathrm{Pl}(U_2 \mid U_3).$$

– $\otimes$ distributes over $\oplus$, more precisely, $a \otimes (\oplus_{i=1}^{\infty} b_i) = \oplus_{i=1}^{\infty}(a \otimes b_i)$ if

  * $(a, b_i), (a, \oplus_{i=1}^{\infty} b_i) \in \mathrm{Dom}(\otimes)$ and
  * $(b_1, b_2, \ldots), (a \otimes b_1, a \otimes b_2, \ldots) \in \mathrm{Dom}(\oplus)$,

  where

$$\begin{aligned} \mathrm{Dom}(\oplus) \;=\; & \{(\mathrm{Pl}(V_1 \mid U), \mathrm{Pl}(V_2 \mid U), \ldots) : \\ & V_1, V_2, \ldots \in \mathcal{F} \text{ are pairwise disjoint and } U \in \mathcal{F}'\}, \end{aligned}$$

  and

$$\begin{aligned} \mathrm{Dom}(\otimes) \;=\; & \{(\mathrm{Pl}(U_1 \mid U_2 \cap U_3), \mathrm{Pl}(U_2 \mid U_3)) : \\ & U_2 \cap U_3 \in F', U_1, U_2, U_3 \in \mathcal{F}\}. \end{aligned}$$

  (The reason that this property is required only for tuples in $\mathrm{Dom}(\oplus)$ and $\mathrm{Dom}(\otimes)$ is discussed shortly.)

– if (a, c), $(b, c) \in \mathrm{Dom}(\otimes)$, $a \otimes c \leq b \otimes c$, and $c \neq \bot$, then $a \leq b$.

To understand the reason for the restriction to $\mathrm{Dom}(\oplus)$ and $\mathrm{Dom}(\otimes)$, consider probability. In that case, D is $[0, 1]$, and we take $\oplus_{i=1}^{\infty} b_i$ to be $\max(\sum_{i=1}^{\infty} b_1, 1)$. It is not too hard to show that the distributive property does not hold in general if $\sum_{i=1}^{\infty} b_i > 1$ (consider, for example $a = 1/2$, $b_1 = b_2 = 2/3$, and $b_i = 0$ for $i \geq 3$);

however, it does hold if $\sum_{i=1}^{\infty} b_i \leq 1$, a property that is guaranteed to hold if there exist sets $V_1, V_2, \ldots$ that are pairwise disjoint and a set U such that $b_i = \mu(V_i \mid U)$ for some probability measure μ.

It can be shown (see [34, 36]) that the constraints on cacps's imply that $\bot$ acts as an identity element for $\oplus$ and that $\top$ acts as an identity element for $\otimes$, just as 0 and 1 do for addition and multiplication, as long as we restrict to tuples in $\mathrm{Dom}(\oplus)$ and $\mathrm{Dom}(\otimes)$, respectively, which is all we care about in our proofs. The constraints also imply that $\mathrm{Pl}(U \mid U) = \top$ for $U \in \mathcal{F}$.

All the plausibility measures we considered earlier can be viewed as examples of cacps's. First, the trivial plausibility measure $\mathrm{Pl}_{\mathrm{triv}}$ is a cacps if we take $\oplus$ to be max and $\otimes$ to be min. A conditional probability space (as just defined) is a cacps simply by defining $\oplus$ as above, so that $\oplus_{i=1}^{\infty} b_i = \max(\sum_{i=1}^{\infty} b_i, 1)$, and taking $\otimes$ to be multiplication. If we have a set $\mathcal{M}$ of probability measures on a space W, we can construct a conditional plausibility measure $\mathrm{Pl}_{\mathcal{M}}$ in essentially the same way that we constructed an unconditional plausibility measure from the set $\mathcal{M}$, so that $\mathrm{Pl}_{\mathcal{M}}(V \mid U)$ is the function $f_{V \mid U}$ from measures in $\mathcal{M}$ to $[0, 1]$ such that $f_{V \mid U}(\mu) = \mu(V \mid U)$ if $\mu(U) \neq 0$, and $f_{V \mid U}(\mu) = *$ (i.e., undefined) if $\mu(U) = 0$. To get a cacps, we simply define $\oplus$ and $\otimes$ pointwise (so that, for example, $f \oplus g$ is that function such that $(f \oplus g)(\mu) = f(\mu) \oplus g(\mu)$). There are subtleties involved in defining the set $\mathcal{F}'$ on which conditioning is defined—in particular, care is required when dealing with sets U such that $\mu(U) > 0$ for some, but not all, of the measures in $\mathcal{M}$. These issues do not affect the results of this paper because we assume that the information sets on which we condition have positive probability, so we ignore them here. See Halpern [36] for more details.

Define a *run-based plausibility system* to be a cacps $(\mathcal{R}, \mathcal{F}, \mathcal{F}', \mathrm{Pl})$. Instead of

requiring that $\mu(\mathcal{R}(\mathcal{K}_i(r,m))) > 0$ as in the probabilistic case, we now require that $\mathcal{R}(\mathcal{K}_i(r,m)) \in \mathcal{F}'$ for all agents i and points (r,m). This requirement guarantees that conditioning on $\mathcal{R}(\mathcal{K}_i(r,m))$ is defined, but is easier to work with than the requirement that $\mu(\mathcal{R}(\mathcal{K}_i(r,m))) > 0$. We can now repeat the Halpern-Tuttle construction to generate standard plausibilistic systems. With this construction, we can explain how the results of Sections 3.2.1, 3.2.2, and 3.2.3 carry over to the more general plausibilistic setting. As a rule of thumb, the results extend to the plausibilistic setting by replacing + and $\times$ consistently in the proofs by $\oplus$ and $\otimes$, but some care is required. We summarize the details here without stating them as formal results; a technical discussion appears in the appendix.

- Proposition 8 generalizes to run-based plausibility systems.

- Theorems 3 and 4 carry over to the plausibilistic setting (with essentially the same proofs) once we define a language for reasoning about plausibility analogous to the language for reasoning about probability, with formulas of the form $\mathrm{Pl}_i(\phi) = \alpha$.

- Proposition 6 generalizes, given a common prior Pl_{cp}, provided that $\otimes$ is commutative. Total secrecy requires that for all points (r,m),

$$\mathrm{Pl}_{cp}(\mathcal{K}_i(r,m) \,|\, \mathcal{PT}(\mathcal{R})) \neq \bot \;\text{ and }\; \mathrm{Pl}_{cp}(\mathcal{K}_j(r,m) \,|\, \mathcal{PT}(\mathcal{R})) \neq \bot;$$

  similarly, synchronous secrecy requires that for all points,

$$\mathrm{Pl}_{cp}(\mathcal{K}_i(r,m) \,|\, \mathcal{PT}(m)) \neq \bot \;\text{ and }\; \mathrm{Pl}_{cp}(\mathcal{K}_j(r,m) \,|\, \mathcal{PT}(m)) \neq \bot.$$

- Proposition 7 generalizes provided that $\otimes$ is commutative and that for all points (r,m) we have $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, \mathcal{R}) \neq \bot$ and $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_j(r,m)) \,|\, \mathcal{R}) \neq \bot$.

- Theorem 5 can be extended using an appropriate definition of adversarial plausibility systems.

These results demonstrate the essential unity of our definitions and theorems in the probabilistic and nonprobabilistic cases, and suggest further generalizations. In particular, it may be worthwhile to consider definitions of secrecy based on representations of uncertainty that are more qualitative than probability is. Such measures might be useful for mitigating the difficulties associated with quantifying probability measures and deciding how nondeterministic choices are resolved.

Chapter 4

Defining Anonymity

4.1 Defining anonymity using knowledge

4.1.1 Revisiting secrecy

In Chapter 3, we looked at requirements of total secrecy in multiagent systems. Total secrecy basically requires that in a system with "classified" and "unclassified" users, the unclassified users should never be able to infer the actions or the local states of the unclassified users. For secrecy, the "what needs to be hidden" component of the requirement is extremely restrictive: total secrecy requires that absolutely everything that a classified user does must be hidden. The "how well does it need to be hidden" component depends on the situation. Our definition of secrecy says that for any *nontrivial* fact ϕ (that is, one that is not already valid) that depends only the state of the classified or high-level agent, the formula $\neg K_j\phi$ must be valid. Semantically, this means that whatever the high-level user does, there exists some run where the low user's view of the system is the same, but the high-level user did something different. Our nonprobabilistic definitions are fairly strong (simply because secrecy requires that so much be hidden). The probabilistic definitions we gave require even more: not only can the agent not learn any new classified fact, but he also cannot learn anything about the probability of any such fact. (In other words, if an agent initially assigns a classified fact ϕ a probability α of being true, he always assigns ϕ that probability.) It would be perfectly natural, and possibly quite interesting, to consider definitions of secrecy that do not require so much to be hidden (e.g., by allowing some classified information to be

declassified [95]), or to discuss definitions that do not require such strong secrecy (e.g., by giving definitions that were stronger than the nonprobabilistic definitions we gave, but not quite so strong as the probabilistic definitions).

4.1.2 Defining anonymity

The basic intuition behind anonymity is that *actions* should be divorced from the *agents* who perform them, for some set of *observers*. Put another way, the information that needs to be hidden is the identity of the agent (or set of agents) who perform a particular action. Who the information needs to be hidden from, that is, which observers, depends on the situation. The most interesting aspect of the definitions of anonymity that we present here will often have to do with how well an agent's identity must be protected.

Throughout this chapter, and when we consider related definitions of anonymity in the following chapter, we use the formula $\theta(i, a)$ to represent "agent i has performed action a, or will perform a in the future."[1] For future reference, let $\delta(i, a)$ represent "agent i has performed action a." Note that $\theta(i, a)$ is a fact about the run: if it is true at some point in a run, it is true at all points in a run (since it is true even if i performs a at some point in the future). On the other hand, $\delta(i, a)$ may be false at the start of a run, and then become true at the point where i performs a.

It is not our goal in this dissertation to provide a "correct" definition of anonymity. We also want to avoid giving an encyclopedia of definitions. Rather,

[1] If we want to consider systems that may crash we may want to consider $\theta'(i, a)$ instead, where $\theta'(i, a)$ represents "agent i has performed action a, or will perform a in the future if the system does not crash." Since issues of failure are orthogonal to the anonymity issues that we focus on here, we consider only the simpler definition.

we give some basic definitions of anonymity to show how our framework can be used. We base our choice of definitions in part on definitions presented in earlier papers, to make clear how our work relates to previous work, and in part on which definitions of anonymity we expect to be useful in practice. We first give an extremely weak definition, but one that nonetheless illustrates the basic intuition behind any definition of anonymity.

Definition 14 *Action a, performed by agent i, is* minimally anonymous *with respect to agent j in the interpreted system $\mathcal{I}$, if $\mathcal{I} \models \neg K_j[\theta(i,a)]$.*

This definition makes it clear what is being hidden ($\theta(i,a)$—the fact that i performs a) and from whom (j). It also describes how well the information is hidden: it requires that j not be sure that i actually performed, or will perform, the action. Note that this is a weak requirement. It might be the case, for example, that agent j is certain that the action was performed either by i, or by at most one or two other agents, thereby making i a "prime suspect." It might also be the case that j is able to place a very high probability on i performing the action, even though he isn't absolutely certain of it. (Agent j might know that there is some slight probability that some other agent i' performed the action, for example.) Nonetheless, it should be the case that for any other definition of anonymity we give, if we want to ensure that i's performing action a is to be kept anonymous as far as observer j is concerned, then i's action should be at least minimally anonymous with respect to j.

Our definition of a being minimally anonymous with respect to j is equivalent to the apparently weaker requirement $\mathcal{I} \models \theta(i,a) \Rightarrow \neg K_j[\theta(i,a)]$, which says that if action a is performed by i, then j does not not know it. Clearly if j never knows that a is performed by i, then j will never know that a is performed by i if i

actually does perform a. To see that the converse holds, it suffices to note that if i does not perform a, then surely $\neg K_j[\theta(i,a)]$ holds. Thus, this definition, like several that will follow, can be viewed as having the form "if i performed a, then j does not know some appropriate fact."

The definition of minimal anonymity also makes it clear how anonymity relates to secrecy, as defined in our earlier work [39]. To explain how, we first need to describe how we defined secrecy in terms of knowledge. Given a system $\mathcal{I}$, say that ϕ is *nontrivial in* $\mathcal{I}$ if $\mathcal{I} \not\models \phi$, and that ϕ *depends only on the local state of agent* i *in* $\mathcal{I}$ if $\mathcal{I} \models \phi \Rightarrow K_i\phi$. Intuitively, ϕ is nontrivial in $\mathcal{I}$ if ϕ could be false in $\mathcal{I}$, and ϕ depends only on i's local state if i always knows whether or not ϕ is true. (It is easy to see that ϕ depends only on the local state of i if $(\mathcal{I}, r, m) \models \phi$ and $r_i(m) = r'_i(m')$ implies that $(\mathcal{I}, r', m') \models \phi$.) According to the definition in [39], agent i *maintains total secrecy with respect to another agent* j *in system* $\mathcal{I}$ if for every nontrivial fact ϕ that depends only on the local state of i, the formula $\neg K_j\phi$ is valid for the system. That is, i maintains total secrecy with respect to j if j does not learn anything new about agent i's state. In general, $\theta(i,a)$ does not depend only on i's local state, because whether i performs a may depend on whether or not i gets a certain message from some other agent i'. On the other hand, if whether or not i performs a depends only on i's protocol, and the protocol is encoded in i's local state, then $\theta(i,a)$ depends only on i's local state. If $\theta(i,a)$ does depend only on i's local state and j did not know all along that i was going to perform action a (i.e., if we assume that $\theta(i,a)$ is nontrivial), then Definition 14 is clearly a special case of the definition of secrecy. In any case, it is in much the same spirit as the definition of secrecy. Essentially, anonymity says that the fact that agent i has or will perform action a must be hidden from j, while total secrecy

says that all facts that depend on agent i must be hidden from j.

Note that this definition of minimal anonymity is different from the one given in an earlier presentation of this work [40]. There, the definition given used $\delta(i,a)$ rather than $\theta(i,a)$. We say that a performed by agent i is minimally δ-anonymous if Definition 14 holds, with $\theta(i,a)$ replaced by $\delta(i,a)$. It is easy to see that minimal anonymity implies minimal δ-anonymity (since $\delta(i,a)$ implies $\theta(i,a)$), but the converse is not true in general. For example, suppose that j gets a signal if i is going to perform action a (before i actually performs the action), but then never finds out exactly when i performs a. Then minimal anonymity does not hold. In runs where i performs a, agent j knows that i will perform a when he gets the signal. On the other hand, minimal δ-anonymity does hold, because j never knows when i performs a. In this situation, minimal anonymity seems to capture our intuitions of what anonymity should mean better than minimal δ-anonymity does.

The next definition of anonymity we give is much stronger. It requires that if some agent i performs an action anonymously with respect to another agent j, then j must think it possible that the action could have been performed by *any* of the agents (except for j). Let $P_j\phi$ be an abbreviation for $\neg K_j \neg \phi$. The operator P_j is the dual of K_j; intuitively, $P_j\phi$ means "agent j thinks that ϕ is possible."

Definition 15 *Action a, performed by agent i, is* totally anonymous *with respect to j in the interpreted system $\mathcal{I}$ if*

$$\mathcal{I} \models \theta(i,a) \Rightarrow \bigwedge_{i' \neq j} P_j[\theta(i',a)].$$

Definition 15 captures the notion that an action is anonymous if, as far as the observer in question is concerned, it could have been performed by anybody in the system.

Again, in the conference version of the paper, we defined total anonymity using $\delta(i, a)$ rather than $\theta(i, a)$. (The same remark holds for all the other definitions of anonymity that we give, although we do not always say so explicitly.) Let total δ-anonymity be the anonymity requirement obtained when $\theta(i, a)$ is replaced by $\delta(i, a)$. It is not hard to show that if agents have perfect recall (which intuitively means that their local state keeps track of all the actions they have performed—see [20] for the formal definition), then total δ-anonymity implies total anonymity. This is not true, in general, without perfect recall, because it might be possible for some agent to know that i will perform action a—and therefore that no other agent will—but forget this fact by the time that i actually performs a. Similarly, total anonymity does not imply total δ-anonymity. To see why, suppose that the agents are numbered $1, \ldots, n$, and that an outside observer knows that if j performs action a, then j will perform it at time j. Then total anonymity may hold even though total δ-anonymity does not. For example, at time 3, although the observer may consider it possible that agent 4 will perform the action (at time 4), he cannot consider it possible that 4 has already performed the action, as required by total δ-anonymity.

Chaum [5] showed that total anonymity could be obtained using DC-nets. Recall that in a DC-net, a group of n users use Chaum's dining cryptographer's protocol (described in the same paper) to achieve anonymous communication. If we model a DC-net as an interpreted multiagent system $\mathcal{I}$ whose agents consist exclusively of agents participating in a single DC-net, then if an agent i sends a message using the DC-net protocol, that action is totally anonymous. (Chaum proves this, under the assumption that any message could be generated by any user in the system.) Note that in the dining cryptographer's example, total anonymity

and δ-total anonymity agree, because who paid is decided before the protocol starts.

It is easy to show that if an action is totally anonymous, then it must be minimally anonymous as well, as long as two simple requirements are satisfied. First, there must be at least 3 agents in the system. (A college student with only one roommate can't anonymously leave out her dirty dishes, but a student with at least two roommates might be able to.) Second, it must be the case that a can be performed only once in a given run of the system. Otherwise, it might be possible for j to think that any agent $i' \neq i$ could have performed a, but for j to *know* that agent i did, indeed, perform a. For example, consider a system with three agents besides j. Agent j might know that all three of the other agents performed action a. In that case, in particular, j knows that i performed a, so action a performed by i is not minimally anonymous with respect to j, but is totally anonymous. We expect that this assumption will typically be met in practice. It is certainly consistent with examples of anonymity given in the literature. (See, for example, [5, 79]). In any case, if it is not met, it is possible to tag occurrences of an action (so that we can talk about the kth time a is performed). Thus, we can talk about the ith occurrence of an action being anonymous. Because the ith occurrence of an action can only happen once in any given run, our requirement is satisfied.

Proposition 9 *Suppose that there are at least three agents in the interpreted system $\mathcal{I}$ and that*

$$\mathcal{I} \models \bigwedge_{i \neq j} \neg[\theta(i,a) \wedge \theta(j,a)].$$

If action a, performed by agent i, is totally anonymous with respect to j, then it is minimally anonymous as well.

Proof: Suppose that action a is totally anonymous. Because there are three agents in the system, there is some agent i' other than i and j, and by total anonymity, $\mathcal{I} \models \theta(i,a) \Rightarrow P_j[\theta(i',a)]$. If $(\mathcal{I},r,m) \models \neg\theta(i,a)$, clearly $(\mathcal{I},r,m) \models \neg K_j[\theta(i,a)]$. Otherwise, $(\mathcal{I},r,m) \models P_j[\theta(i',a)]$ by total anonymity. Thus, there exists a point (r',m') such that $r'_j(m') = r_j(m)$ and $(\mathcal{I},r',m') \models \theta(i',a)$. By our assumption, $(\mathcal{I},r',m') \models \neg\theta(i,a)$, because $i \neq i'$. Therefore, $(\mathcal{I},r,m) \models \neg\mathcal{K}_j[\theta(i,a)]$. It follows that a is minimally anonymous with respect to j. □

Definitions 14 and 15 are conceptually similar, even though the latter definition is much stronger. Once again, there is a set of formulas that an observer is not allowed to know. With the earlier definition, there is only one formula in this set: $\theta(i,a)$. As long as j doesn't know that i performed action a, this requirement is satisfied. With total anonymity, there are more formulas that j is not allowed to know: they take the form $\neg\theta(i',a)$. Before, we could guarantee only that j did not know that i did the action; here, for many agents i', we guarantee that j does not know that i' did *not* do the action. The definition is made slightly more complicated by the implication, which restricts the conditions under which j is not allowed to know $\neg\theta(i',a)$. (If i didn't actually perform the action, we don't care what j thinks, since we are concerned only with anonymity with respect to i.) But the basic idea is the same.

Note that total anonymity does *not* necessarily follow from total secrecy, because the formula $\neg\theta(i',a)$, for $i' \neq i$, does not, in general, depend only on the local state of i. It is therefore perfectly consistent with the definition of total secrecy for j to learn this fact, in violation of total anonymity. (Secrecy, of course, does not follow from anonymity, because secrecy requires that many more facts be hidden than simply whether i performed a given action.)

Total anonymity is a very strong requirement. Often, an action will not be totally anonymous, but only anonymous up to some set of agents who could have performed the action. This situation merits a weaker definition of anonymity. To be more precise, let I be the set of all agents of the system and suppose that we have some set $I_A \subseteq I$—an "anonymity set," using the terminology of Chaum [5] and Pfitzmann and Köhntopp [68]—of agents who can perform some action. We can define anonymity in terms of this set.

Definition 16 *Action a, performed by agent i, is* anonymous up to $I_A \subseteq I$ *with respect to j if*

$$\mathcal{I} \models \theta(i,a) \Rightarrow \bigwedge_{i' \in I_A} P_j[\theta(i',a)].$$

In the anonymous message-passing system Herbivore [30], users are organized into *cliques* $C_1, \ldots, C_n$, each of which uses the dining cryptographers protocol [5] for anonymous message-transmission. If a user wants to send an anonymous message, she can do so through her clique. Herbivore claims that any user i is able to send a message anonymously up to C_j, where $i \in C_j$. As the size of a user's clique varies, so does the strength of the anonymity guarantees provided by the system.

In some situations, it is not necessary that there be a fixed anonymity set, as in Definition 16. It suffices that, at all times, there *exists* some anonymity set with at least, say, k agents. This leads to a definition of k-anonymity.

Definition 17 *Action a, performed by agent i, is* k-anonymous *with respect to j if*

$$\mathcal{I} \models \theta(i,a) \Rightarrow \bigvee_{\{I_A : |I_A| = k\}} \bigwedge_{i' \in I_A} P_j[\theta(i',a)].$$

This definition says that at any point j must think it possible that any of at least k agents might perform, or have performed, the action. Note that the set of

k agents might be different in different runs, making this condition strictly weaker than anonymity up to a particular set of size k.

A number of systems have been proposed that provide k-anonymity for some k. In the anonymous communications network protocol recently proposed by von Ahn, Bortz, and Hopper [1], users can send messages with guarantees of k-anonymity. In the system P^5 (for "Peer-to-Peer Personal Privacy Protocol") [82], users join a logical broadcast tree that provides anonymous communication, and users can choose what level of k-anonymity they want, given that k-anonymity for a higher value of k makes communication more inefficient. Herbivore [30] provides anonymity using cliques of DC-nets. If the system guarantees that the cliques all have a size of at least k, so that regardless of clique composition, there are at least k users capable of sending any anonymous message, then Herbivore guarantees k-anonymity.

4.1.3 A more detailed example: dining cryptographers

A well-known example of anonymity in the computer security literature is Chaum's "dining cryptographers problem" [5]. In the original description of this problem, three cryptographers sit down to dinner and are informed by the host that someone has already paid the bill anonymously. The cryptographers decide that the bill was paid either by one of the three people in their group, or by an outside agency such as the NSA. They want to find out which of these two situations is the actual one while preserving the anonymity of the cryptographer who (might have) paid. Chaum provides a protocol that the cryptographers can use to solve this problem. To guarantee that it works, however, it would be nice to check that anonymity conditions hold. Assuming we have a system that includes a set of three cryptographer agents $C = \{0, 1, 2\}$, as well as an outside observer agent o,

the protocol should guarantee that for each agent $i \in C$, and each agent $j \in C-\{i\}$, the act of paying is anonymous up to $C - \{j\}$ with respect to j. For an outside observer o, i.e., an agent other than one of three cryptographers, the protocol should guarantee that for each agent $i \in C$, the protocol is anonymous up to C with respect to o. This can be made precise using our definition of anonymity up to a set.

Because the requirements are symmetric for each of the three cryptographers, we can describe the anonymity specification compactly by naming the agents using modular arithmetic. We use $\oplus$ to denote addition mod 3. Let the interpreted system $(\mathcal{I} = (\mathcal{R}, \pi)$ represent the possible runs of one instance of the dining cryptographers protocol, where the interpretation π interprets formulas of the form $\theta(i, \text{“paid”})$ in the obvious way. The following knowledge-based requirements comprise the anonymity portion of the protocol's specification, for each agent $i \in C$:

$$\begin{aligned} \mathcal{I} \models \theta(i, \text{“paid”}) \quad &\Rightarrow P_{i\oplus 1}\theta(i \oplus 2, \text{“paid”}) \wedge \; P_{i\oplus 2}\theta(i \oplus 1, \text{“paid”}) \\ &\wedge \; P_o\theta(i \oplus 1, \text{“paid”}) \wedge \; P_o\theta(i \oplus 2, \text{“paid”}). \end{aligned}$$

In other words, if cryptographer i paid, then both of the other cryptographers must think it possible that the third cryptographer could have paid. In addition, an outside observer must think it possible that any of the three cryptographers could have paid.

4.2 Probabilistic variants of anonymity

4.2.1 Probabilistic anonymity

All of the definitions presented in Section 4.1 were nonprobabilistic. As with secrecy, nonprobabilistic definitions of anonymity are, in some ways, quite weak. For

all the definitions we gave, it was necessary only that observers think it *possible* that multiple agents could have performed the anonymous action. However, an event that is possible may nonetheless be extremely unlikely. Consider our definition of total anonymity (Definition 15). It states that an action performed by i is totally anonymous if the observer j thinks it could have been performed by any agent other than j. This may seem like a strong requirement, but it can look quite weak when probabilities are involved. Suppose, for example that we have 102 agents, that j can determine that i performed action a with probability 0.99, and that j believes that each of the other agents performed action a with probability 0.0001. In this case, agent i might not be very happy with the guarantees provided by total anonymity. Of course, the appropriate notion of anonymity will depend on the application: i might be content to know that no agent can *prove* that she performed the anonymous action. In that case, it might suffice for the action to be only minimally anonymous. However, in many other cases, an agent might want a more quantitative, probabilistic guarantee that it will be considered reasonably likely that other agents could have performed the action.

We focus here on definitions of anonymity given with respect to interpreted standard probability systems, as defined in Section 2.2. We write $\mathcal{I} = (\mathcal{R}, \mu, \pi)$ to denote an interpreted probability system. (The probability assignment $\mathcal{PR}$ is determined by μ according to the Halpern-Tuttle construction.)

It is straightforward to define probabilistic notions of anonymity in probabilistic systems. We can think of Definition 14, for example, as saying that j's probability that i performs the anonymous action a must be less than 1 (assuming that every nonempty set has positive probability). This can be generalized by specifying some $\alpha \leq 1$ and requiring that the probability of $\theta(i, a)$ be less than α.

Definition 18 *Action a, performed by agent i, is* α-anonymous *with respect to agent j if* $\mathcal{I} \models \theta(i,a) \Rightarrow \Pr_j[\theta(i,a)] < \alpha$.

Note that if we replace $\theta(i,a)$ by $\delta(i,a)$ in Definition 18, the resulting notion might not be well defined. The problem is that the set

$$\{(r',m') \in \mathcal{K}_i(r,m) : (\mathcal{I},r',m') \models \delta(i,a)\}$$

may not be measurable; it may not have the form $\mathcal{K}_i(r,m)(\mathcal{S})$ for some $\mathcal{S} \subseteq \mathcal{R}$. The problem does not arise if $\mathcal{I}$ is a *synchronous* system (in which case i knows that time, and all the points in $\mathcal{K}_i(r,m)$ are of the form (r',m)), but it does arise if $\mathcal{I}$ is asynchronous. We avoid this technical problem by working with $\theta(i,a)$ rather than $\delta(i,a)$.

Definition 18, unlike Definition 14, includes an implication involving $\theta(i,a)$. It is easy to check that Definition 14 does not change when such an implication is added; intuitively, if $\theta(i,a)$ is false then $\neg K_j[\theta(i,a)]$ is trivially true. Definition 18, however, would change if we removed the implication, because it might be possible for j to have a high probability of $\theta(i,a)$ even though it isn't true. We include the implication because without it, we place constraints on what j thinks about $\theta(i,a)$ even if i has not performed the action a and will not perform it in the future. Such a requirement, while interesting, seems more akin to "unsuspectibility" than to anonymity.

Two of the notions of probabilistic anonymity considered by Reiter and Rubin [71] in the context of their Crowds system can be understood in terms of α-anonymity. Reiter and Rubin say that a sender has *probable innocence* if, from an observer's point of view, the sender "appears no more likely to be the originator than to not be the originator." This is simply 0.5-anonymity. (Under

reasonable assumptions, Crowds provides 0.5-anonymity for Web requests.) Similarly, a sender has *possible innocence* if, from an observer's point of view, "there is a nontrivial probability that the real sender is someone else." This corresponds to minimal anonymity (as defined in Section 4.1.2), or to ϵ-anonymity for some nontrivial value of ϵ.

It might seem at first that Definition 18 should be the only definition of anonymity we need: as long as j's probability of i performing the action is low enough, i should have nothing to worry about. However, with further thought, it is not hard to see that this is not the case.

Consider a scenario where there are 1002 agents, and where $\alpha = 0.11$. Suppose that the probability, according to Alice, that Bob performs the action is .1, but that her probability that any of the other 1000 agents performs the action is 0.0009 (for each agent). Alice's probability that Bob performs the action is small, but her probability that anyone else performs it is more than three orders of magnitude smaller. Bob is obviously the prime suspect.

This concern was addressed by Serjantov and Danezis [80] in their paper on information-theoretic definitions of anonymity. They consider the probability that each agent in an anonymity set is the sender of some anonymous message, and use entropy to quantify the amount of information that the system is leaking; Diaz et al. [15] and Danezis [11] use similar techniques. In this dissertation we do not consider quantitative measurements of anonymity, but we do agree that it is worthwhile to consider stronger notions of anonymity than the nonprobabilistic definitions, or even α-anonymity, can provide. We hope to examine quantitative definitions in future work.

The next definition strengthens Definition 18 in the way that Definition 15

strengthens Definition 14. It requires that no agent in the anonymity set be a more likely suspect than any other.

Definition 19 *Action a, performed by agent i, is* strongly probabilistically anonymous up to I_A *with respect to agent j if for each $i' \in I_A$,*

$$\mathcal{I} \models \theta(i,a) \Rightarrow \Pr_j[\theta(i,a)] = \Pr_j[\theta(i',a)].$$

Depending on the size of I_A, this definition can be extremely strong. It does not state simply that for all agents in I_A, the observer must think it is reasonably likely that the agent could have performed the action; it also says that the observer's probabilities must be the same for each such agent. Of course, we could weaken the definition somewhat by not requiring that all the probabilities be equal, but by instead requiring that they be approximately equal (i.e., that their difference be small or that their ratio be close to 1). Reiter and Rubin [71], for example, say that the sender of a message is *beyond suspicion* if she "appears no more likely to be the originator of that message than any other potential sender in the system." In our terminology, i is beyond suspicion with respect to j if for each $i' \in I_A$,

$$\mathcal{I} \models \theta(i,a) \Rightarrow \Pr_j[\theta(i,a)] \leq \Pr_j[\theta(i',a)].$$

This is clearly weaker than strong probabilistic anonymity, but still a very strong requirement, and perhaps more reasonable, too. Our main point is that a wide variety of properties can be expressed clearly and succinctly in our framework.

4.2.2 Conditional anonymity

While we have shown that many useful notions of anonymity—including many definitions that have already been proposed—can be expressed in our framework,

we claim that there are some important intuitions that have not yet been captured. Suppose, for example, that someone makes a $5,000,000 donation to Cornell University. It is clearly not the case that everyone is equally likely, or even almost equally likely, to have made the donation. Of course, we could take the anonymity set I_A to consist of those people who might be in a position to make such a large donation, and insist that they all be considered equally likely. Unfortunately, even that is unreasonable: a priori, some of them may already have known connections to Cornell, and thus be considered far more likely to have made the donation. All that an anonymous donor can reasonably expect is that nothing an observer learns from his interactions with the environment (e.g., reading the newspapers, noting when the donation was made, etc.) will give him more information about the identity of the donor than he already had.

For another example, consider a conference or research journal that provides anonymous reviews to researchers who submit their papers for publication. It is unlikely that the review process provides anything like α-anonymity for a small α, or strongly probabilistic anonymity up to some reasonable set. When a preliminary version of this work, for example, was accepted by the Computer Security Foundations Workshop, the acceptance notice included three reviews that were, in our terminology, anonymous up to the program committee. That is, any one of the reviews we received could have been written by any of the members of the program committee. However, by reading some of the reviews, we were able to make fairly good guesses as to which committee members had provided which reviews, based on our knowledge of the specializations of the various members, and based on the content of the reviews themselves. Moreover, we had a fairly good idea of which committee members would provide reviews of the paper even before we received

the reviews. Thus, it seems unreasonable to hope that the review process would provide strong probabilistic anonymity (up to the program committee), or even some weaker variant of probabilistic anonymity. Probabilistic anonymity would require the reviews to convert our prior beliefs, according to which some program committee members were more likely than others to be reviewers of the paper, to posterior beliefs according to which all program committee members were equally likely. This does not seem at all reasonable. However, the reviewers might hope that that the process did not give us any more information than we already had.

In the definitions of secrecy given in the previous chapter, we tried to capture the intuition that, when an unclassified user interacts with a secure system, she does not learn anything about any classified user that she didn't already know. We did this formally by requiring that, for any three points (r, m), (r', m'), and (r'', m''),

$$\mu_{(r,m,j)}(\mathcal{K}_i(r'', m'')) = \mu_{(r',m',j)}(\mathcal{K}_i(r'', m'')). \tag{4.1}$$

That is, whatever the unclassified user j sees, her probability of any particular classified state will remain unchanged.

When defining anonymity, we are not concerned with protecting all information about some agent i, but rather the fact that i performs some particular action a. Given an interpreted system $\mathcal{I} = (\mathcal{R}, \pi, \mu)$ and a formula ϕ, let $e_r(\phi)$ consist of the set of runs r such that ϕ is true at some point in r, and let $e_p(\phi)$ be the set of points where ϕ is true. That is

$$e_r(\phi) \triangleq \{r : \exists m((\mathcal{I}, r, m) \models \phi)\},$$
$$e_p(\phi) \triangleq \{(r, m) : (\mathcal{I}, r, m) \models \phi\}.$$

The most obvious analogue to (4.1) is the requirement that, for all points (r, m)

and (r', m'),

$$\mu_{(r,m,j)}(e_p(\theta(i,a))) = \mu_{(r',m',j)}(e_p(\theta(i,a))).$$

This definition says that j never learns anything about the probability that i performs performs a: she always ascribes the same probability to this event. In the context of our anonymous donation example, this would say that the probability (according to j) of i donating \$5,000,000 to Cornell is the same at all times.

The problem with this definition is that it does not allow j to learn that *someone* donated \$5,000,000 to Cornell. That is, before j learned that someone donated \$5,000,000 to Cornell, j may have thought it was unlikely that anyone would donate that much money to Cornell. We cannot expect that j's probability of i donating \$5,000,000 would be the same both before and after learning that someone made a donation. We want to give a definition of conditional anonymity that allows observers to learn that an action has been performed, but that protects—as much as possible, given the system—the fact that some particular agent performs the action. If, on the other hand, the anonymous action has not been performed, then the observer's probabilities do not matter.

Suppose that i wants to perform action a, and wants conditional anonymity with respect to j. Let $\theta(\overline{j}, a)$ represent the fact that a has been performed by some agent other than j, that is, $\theta(\overline{j}, a) \triangleq \vee_{i' \neq j} \theta(i', a)$. The definition of conditional anonymity says that j's prior probability of $\theta(i, a)$ given $\theta(\overline{j}, a)$ must be the same as his posterior probability of $\theta(i, a)$ at points where j knows $\theta(\overline{j}, a)$, i.e., at points where j knows that someone other than j has performed (or will perform) a. Let $\alpha = \mu(e_r(\theta(i, a)) \,|\, e_r(\theta(\overline{j}, a)))$. This is the prior probability that i has performed a, given that somebody other than j has. Conditional anonymity says that at any point where j knows that someone other than j performs a, j's probability

of $\theta(i,a)$ must be α. In other words, j shouldn't be able to learn anything more about who performs a (except that somebody does) than he know before he began interacting with the system in the first place.

Definition 20 *Action a, performed by agent i, is* conditionally anonymous *with respect to j in the interpreted probability system $\mathcal{I}$ if*

$$\mathcal{I} \models K_j\theta(\overline{j},a) \Rightarrow \Pr_j(\theta(i,a)) = \mu(e_r(\theta(i,a)) \mid e_r(\theta(\overline{j},a))).$$

Note that if only one agent ever performs a, then a is trivially conditionally anonymous with respect to j, but may not be minimally anonymous with respect to j. Thus, conditional anonymity does not necessarily imply minimal anonymity.

In Definition 20, we implicitly assumed that agent j was allowed to learn that someone other than j performed action a; anonymity is intended to hide *which* agent performed a, given that somebody did. More generally, we believe that we need to consider anonymity with respect to what an observer is allowed to learn. We might want to specify, for example, that an observer is allowed to know that a donation was made, and for how much, or to learn the contents of a conference paper review. The following definition lets us do this formally.

Definition 21 *Action a, performed by agent i, is* conditionally anonymous *with respect to j and ϕ in the interpreted probability system $\mathcal{I}$ if*

$$\mathcal{I} \models K_j\phi \Rightarrow \Pr_j(\theta(i,a)) = \mu(e_r(\theta(i,a)) \mid e_r(\phi)).$$

Definition 20 is clearly the special case of Definition 21 where $\phi = \theta(\overline{j},a)$. Intuitively, both of these definitions say that once an observer learns some fact ϕ connected to the fact $\theta(i,a)$, we require that she doesn't learn anything else that might change her probabilities of $\theta(i,a)$.

4.2.3 Example: probabilistic dining cryptographers

Returning the dining cryptographers problem, suppose that it is well-known that one of the three cryptographers at the table is much more generous than the other two, and therefore more likely to pay for dinner. Suppose, for example, that the probability measure on the set of runs where the generous cryptographer has paid is 0.8, given that one of the cryptographers paid for dinner, and that it is 0.1 for each of the other two cryptographers. Conditional anonymity for each of the three cryptographers with respect to an outside observer means that when such observer learns that one of the cryptographers has paid for dinner, his probability that any of the three cryptographers paid should remain 0.8, 0.1, and 0.1. If the one of the thrifty cryptographers paid, the generous cryptographer should think that there is a probability of $0.5 = 0.1/(0.1 + 0.1)$ that either of the others paid. Likewise, if the generous cryptographer paid, each of the others should think that there is a probability of $0.8/(0.8 + 0.1)$ that the generous cryptographer paid and a probability of $0.1/(0.8 + 0.1)$ that the other thrifty cryptographer paid. We can similarly calculate all the other relevant probabilities.

More generally, suppose that $(\mathcal{R}, \mu, \pi)$ is an interpreted probabilistic system representing instances of the dining cryptographers protocol, where the interpretation π once again interprets formulas of the form $\theta(i, \text{“paid”})$ and $\theta(\overline{j}, \text{“paid”})$ in the obvious way, and where the formula γ is true if one of the cryptographers paid. (That is, γ is equivalent to $\bigvee_{i\in\{0,1,2\}} \theta(i, \text{“paid”})$.) For any cryptographer $i \in \{0, 1, 2\}$, let $\alpha(i)$ be the prior probability that i paid, given that somebody else did. That is, let

$$\alpha(i) \triangleq \mu(e_r(\theta(i, \text{“paid”})) \,|\, e_r(\gamma)).$$

In the more concrete example given above, if 0 is the generous cryptographer, we

would have $\alpha(0) = 0.8$ and $\alpha(1) = \alpha(2) = 0.1$.

For the purposes of conditional probability with respect to an agent j, we are interested in the probability that some agent i paid, given that somebody other than j paid. Formally, for $i \neq j$, let

$$\alpha(i,j) \triangleq \mu(e_r(\theta(i, \text{“paid”})) \,|\, e_r(\theta(\overline{j}, \text{“paid”}))).$$

If an observer o is not one of the three cryptographers, than o didn't pay, and we have $\alpha(i,o) = \alpha(i)$. Otherwise, if $i, j \in \{0,1,2\}$, we can use conditioning to compute $\alpha(i,j)$:

$$\alpha(i,j) = \frac{\alpha(i)}{\alpha(j \oplus 1) + \alpha(j \oplus 2)}.$$

(Once again, we make our definitions and requirements more compact by using modular arithmetic, where $\oplus$ denotes addition mod 3.)

The following formula captures the requirement of conditional anonymity in the dining cryptographer's protocol, for each cryptographer i, with respect to the other cryptographers and any outside observers.

$$\begin{aligned}
\mathcal{I} \models \quad & \left[K_{i\oplus 1}\theta(\overline{i \oplus 1}, \text{“paid”}) \Rightarrow \Pr_{i\oplus 1}(\theta(i, \text{“paid”})) = \alpha(i, i \oplus 1)\right] \wedge \\
& \left[K_{i\oplus 2}\theta(\overline{i \oplus 2}, \text{“paid”}) \Rightarrow \Pr_{i\oplus 2}(\theta(i, \text{“paid”})) = \alpha(i, i \oplus 2)\right] \wedge \\
& \left[K_o\theta(\overline{o}, \text{“paid”}) \Rightarrow \Pr_o(\theta(i, \text{“paid”})) = \alpha(i,o)\right].
\end{aligned}$$

Chaum's original proof that the dining cryptographers protocol provides anonymity actually proves conditional anonymity in this general setting. Note that if the probability that one of the cryptographers will pay is 1, that cryptographer will have conditional anonymity even though he doesn't even have minimal anonymity.

4.2.4 Other uses for probability

In the previous two subsections, we have emphasized how probability can be used to obtain definitions of anonymity stronger than those presented in Section 4.1. However, probabilistic systems can also be used to define interesting ways of weakening those definitions. Real-world anonymity systems do not offer absolute guarantees of anonymity such as those those specified by our definitions. Rather, they guarantee that a user's anonymity will be protected *with high probability*. In a given run, a user's anonymity might be protected or corrupted. If the probability of the event that a user's anonymity is corrupted is very small, i.e., the set of runs where her anonymity is not protected is assigned a very small probability by the measure μ, this might be enough of a guarantee for the user to interact with the system.

Recall that we said that i maintains total anonymity with respect to j if the fact $\phi = \theta(i,a) \Rightarrow \bigwedge_{i' \neq j} P_j[\theta(i',a)]$ is true at every point in the system. Total anonymity is compromised in a run r if at some point (r,m), $\neg\phi$ holds. Therefore, the set of runs where total anonymity is compromised is simply $e_r(\neg\phi)$, using the notation of the previous section. If $\mu(e_r(\neg\phi))$ is very small, then i maintains total anonymity with very high probability. This analysis can obviously be extended to all the other definitions of anonymity given in previous sections.

Bounds such as these are useful for analyzing real-world systems. The Crowds system [71], for example, uses randomization when routing communication traffic, so that anonymity is protected with high probability. The probabilistic guarantees provided by Crowds were analyzed formally by Shmatikov [83], using a probabilistic model checker, and he demonstrates how the anonymity guarantees provided by the Crowds system change as more users (who may be either honest or corrupt) are added to the system. Shmatikov uses a temporal probabilistic logic to express

probabilistic anonymity properties, so these properties can be expressed in our system framework. (It is straightforward to give semantics to temporal operators in systems; see [20].) In any case, Shmatikov's analysis of a real-world anonymity system is a useful example of how the formal methods that we advocate can be used to specify and verify properties of real-world systems.

Chapter 5

Related Definitions of Secrecy and Anonymity

5.1 Related definitions of secrecy

We are certainly not the first to discuss formal definitions of secrecy: many definitions have been proposed over the last two decades. One reason for this is that researchers have sought an "ideal" definition of security that has a variety of useful properties (such as verifiability and composability). While we certainly agree that verifiability and composability are important properties, we believe that the intuition behind secrecy should be isolated from stronger properties that happen to imply secrecy—especially when we have to worry about subtle issues such as probability and nondeterminism.

In this section we consider how our definitions relate to other information-flow conditions. We show in particular how they can capture work that has been done in the synchronous setting, the asynchronous setting, and the probabilistic setting. Because there are literally dozens of papers that have, in one way or another, defined notions of secrecy or confidentiality, this section is in no way meant to be comprehensive or representative. Rather, we have chosen examples that inspired our definitions, or examples for which our definitions give some insight. In light of our earlier comments, we also focus on definitions that have tried to capture the essence of secrecy rather than notions that have been more concerned with issues like composability and verification.

One important strand of literature to which we do not compare our work di-

rectly here is the work on defining information flow and noninterference using process algebras related to CCS and CSP [21, 22, 73, 74]. Although we believe that the intuitions behind many of these definitions are closely related to our notions of secrecy, a careful discussion of this issue would take us too far afield. In future work we hope to consider the issue in detail, by describing how processes can be translated to the runs-and-systems framework in a way that captures their semantics and then showing how some of the process-algebraic definitions can be recast as examples of secrecy. In Section 5.2.2 we give one instance of such a translation: we show how definitions of anonymity given using CSP by Schneider and Sidiropoulos [79] can be captured in the runs-and-systems framework.

5.1.1 Secrecy in trace systems

Many papers in computer security define information-flow conditions such as secrecy and noninterference using trace-based models. Traces are usually defined as sequences of input and output events, where each event is associated with some agent (either as an input that she provides or an output that she sees). However, there have been some subtle differences among the trace-based models. In some cases, infinite traces are used; in others, the traces are finite. Similarly, some models assume that the underlying systems are synchronous while others do not. Although asynchronous system models have been more common, we first consider synchronous trace-based systems.

Both McLean [58] and Wittbold and Johnson [93] present their definitions of security in the context of synchronous input/output traces. These traces are essentially restricted versions of runs. Here we consider a slightly simplified version of McLean's framework and describe two well-known noninterference properties

within the framework.

Let $\mathbf{In}(H)$ be a set of possible high input values, let $\mathbf{In}(L)$ be a set of possible low input values, let $\mathbf{Out}(H)$ be a set of possible high output values, and $\mathbf{Out}(L)$ be a set of possible low output values. (We assume that these sets are pairwise disjoint and finite.) Write $\mathbf{Ev}(L) = \mathbf{In}(L) \cup \mathbf{Out}(L)$, $\mathbf{Ev}(H) = \mathbf{In}(H) \cup \mathbf{Out}(H)$, and $\mathbf{Ev} = \mathbf{Ev}(L) \cup \mathbf{Ev}(H)$.

A tuple $\gamma = \langle l_i, h_i, l_o, h_o \rangle$ (with $l_i \in \mathbf{In}(L), h_i \in \mathbf{In}(H)$, $l_o \in \mathbf{Out}(L)$, and $h_o \in \mathbf{Out}(H)$) represents a snapshot of a system at a given point in time; it describes the input provided to the system by a low agent L and a high agent H, and the output sent by the system to L and H. A *synchronous trace* $t = \langle \gamma_1, \gamma_2, \ldots \rangle$ is an infinite sequence of such tuples. It represents an infinite execution sequence of the entire system by describing the input/output behavior of the system at any given point in time. (The traces are said to be synchronous because the input and output values are specified for each agent at each time step, and both agents can infer the time simply by looking at the number of system outputs they have seen.) A *synchronous trace system* is a set Σ of synchronous traces, representing the possible execution sequences of the system.

In a synchronous trace system, the local state of an agent can be defined using a *restriction* function on traces. Given a set $E \subseteq \mathbf{Ev}$, $t \restriction E$ gives us the trace t with all elements not in E removed. For example, if

$$t = \langle \langle l_i^{(1)}, h_i^{(1)}, l_o^{(1)}, h_o^{(1)} \rangle, \langle l_i^{(2)}, h_i^{(2)}, l_o^{(2)}, h_o^{(2)} \rangle, \ldots \rangle,$$

then

$$t \restriction \mathbf{Ev}(L) = \langle \langle l_i^{(1)}, l_o^{(1)} \rangle, \langle l_i^{(2)}, l_o^{(2)} \rangle, \ldots \rangle.$$

Similarly, $t \restriction \mathbf{Ev}(H)$ contains only high events, and $t \restriction \mathbf{In}(H)$ contains only high input events. For brevity we write $t \restriction L$ and $t \restriction H$ as shorthand for $t \restriction \mathbf{Ev}(L)$

and $t \restriction \mathbf{Ev}(H)$, respectively. Given a restriction set E, we denote the restriction function $(\cdot \restriction E)$ as $\restriction_E$ (as a notational convenience).

Given a trace $t = \langle \gamma_1, \gamma_2, \ldots \rangle$, the *length-k prefix* of t is $t_k = \langle \gamma_1, \gamma_2, \ldots, \gamma_k \rangle$, that is, the finite sequence containing the first k state tuples of the trace t. Trace restriction applies to trace prefixes in the obvious way.

It is easy to see that synchronous trace systems can be viewed as systems in the multiagent systems framework. Given a trace t, we can define the run r^t such that $r^t(m) = (t_m \restriction L, t_m \restriction H)$. (For simplicity, we have omitted the environment state from the global state in this construction, since it plays no role.) Given a synchronous trace system Σ, let $\mathcal{R}(\Sigma) = \{r^t : t \in \Sigma\}$. It is easy to check that $R(\Sigma)$ is synchronous, and that both agents L and H have perfect recall.

McLean defines a number of notions of secrecy in his framework. We consider two of the best known here: *separability* [58] and *generalized noninterference* [56]. Separability, as its name suggests, ensures secrecy between the low and high agents, whereas generalized noninterference ensures that the low agent is unable to know anything about high input behavior.

Definition 22 *A synchronous trace system Σ satisfies* separability *if, for every pair of traces $t, t' \in \Sigma$, there exists a trace $t'' \in \Sigma$ such that $t'' \restriction L = t \restriction L$ and $t'' \restriction H = t' \restriction H$.*

Definition 23 *A synchronous trace system Σ satisfies* generalized noninterference *if, for every pair of traces $t, t' \in \Sigma$, there exists a trace $t'' \in \Sigma$ such that $t'' \restriction L = t \restriction L$ and $t'' \restriction \mathbf{In}(H) = t' \restriction \mathbf{In}(H)$.*

These definitions are both special cases of nondeducibility, as discussed in Section 3.1.1: take the set of worlds W to be Σ, the information function g to be $\restriction_L$,

and the information function h to be $\restriction_H$ (for separability) and $\restriction_{\mathbf{In}(H)}$ (for generalized noninterference). (It is not difficult to see that if the information functions g and h are limited to trace restriction functions, then nondeducibility is essentially equivalent in expressive power to *selective interleaving functions*, the mechanism for defining security properties introduced by McLean [58].) In our framework, separability essentially corresponds to synchronous secrecy, whereas generalized noninterference corresponds to synchronous $\restriction_{\mathbf{In}(H)}$-secrecy. The following proposition makes this precise. Let f_{hi} be the information function that extracts a high input trace prefix from a point in exactly the same way that $\restriction_{\mathbf{In}(H)}$ extracts it from the infinite trace.

Proposition 10 *If a synchronous trace system Σ satisfies separability (resp., generalized noninterference), then H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

The converse to Proposition 10 is not quite true. There is a subtle but significant difference between McLean's framework and ours. McLean works with *infinite* traces; separability and generalized noninterference are defined with respect to *traces* rather than sets of *points* (i.e., trace prefixes). To see the impact of this, consider a system Σ where the high agent inputs either infinitely many 0s or infinitely many 1s. The output to the low agent is always finitely many 0s followed by infinitely 1s, except for a single trace where the high agent inputs infinitely many 0s and the low agent inputs infinitely many 0s. Thus, the system consists of the following traces, where we have omitted the low inputs since they do not

matter, and the high outputs, which can taken to be constant:

$$(0^k 1^\infty, 0^\infty), \quad k = 0, 1, 2, 3, \ldots$$
$$(0^k 1^\infty, 1^\infty), \quad k = 0, 1, 2, 3, \ldots$$
$$(0^\infty, 0^\infty).$$

In the system $\mathcal{R}(\Sigma)$, H maintains synchronous secrecy and thus synchronous f_{hi}-secrecy with respect to L, because by looking at any finite trace prefix, L cannot tell whether the high inputs have been 0s or 1s. However, Σ does not satisfy separability or generalized interference. If L "sees" infinitely many 0s, he immediately knows that the high inputs have been 0s. This seems unreasonable. After all, agents only makes observations at finite points in time.

Note that if t is a trace where the low outputs are all 0s and the high inputs are all 1s, each finite prefix of the trace t is a prefix of a trace in Σ, even though t is not in Σ. This turns out to be the key reason that the system satisfies synchronous secrecy but not separability.

Definition 24 *A synchronous trace system Σ is* limit closed *[16] if, for all synchronous traces t, we have $t \in \Sigma$ iff for every time k there exists a trace $t' \in \Sigma$ such that $t'_k = t_k$.*

Under the assumption of limit closure, we do get the converse to Proposition 10.

Proposition 11 *A limit-closed synchronous trace system Σ satisfies separability (resp. generalized noninterference) iff H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

While we believe that it is unreasonable in general to assume that an agent's view includes the entire run (as McLean's definitions implicitly do), these re-

sults nonetheless demonstrate the close connection between our definition of synchronous f-secrecy and security properties such as separability and generalized noninterference.

Up to now we have considered a synchronous trace model, where the input and output events of high and low users occur in lockstep. However, many trace-based definitions of security are given in an asynchronous setting. We consider a number of definitions of secrecy in this setting. For uniformity we use terminology similar to that of Mantel [54], who has carefully compiled a variety of well-known trace-based properties into a single framework.

In Mantel's framework, traces are not infinite sequences of input/output value tuples, but finite sequences of input/output events. As before, let the set **Ev** consist of high and low input and output events. Let α range over elements of **Ev**. If $l, l' \in \mathbf{Ev}(L)$ and $h, h' \in \mathbf{Ev}(H)$, a possible system trace could be

$$t = \langle l, h, l, h', h', l', l', l, h \rangle.$$

We define trace restriction as before. Given t above, for example, we have

$$t \upharpoonright L = \langle l, l, l', l', l \rangle.$$

Note that because asynchronous traces are sequences of events rather than tuples, restriction ignores high events altogether. This means that a low view of the system may remain completely unchanged even as many high input events occur.

An *asynchronous trace system* is a set of traces that is closed under trace prefixes. There is a straightforward way of associating with each system a set of runs. A set T of traces is *run-like* if, for all traces t_1 and t_2 in T, either t_1 is a prefix of t_2 or t_2 is a prefix of t_1. Intuitively, a run corresponds to a maximal run-like set of traces. More formally, let T be a maximal run-like set of traces. Note that if T

is infinite, then for all $n \geq 0$ there exists exactly one trace in T of length n (where the length of $\langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ is n); if T is finite, then there is some $N \geq 0$ such that T has exactly one trace of length n for all $n \leq N$. If T is infinite, let the run r^T be such that $r^T(m) = \langle t^m \restriction L, t^m \restriction H \rangle$, where t^m is the unique trace in T of length m. If T is finite, let r^T be such that $r^T(m) = \langle t^m \restriction L, t^m \restriction H \rangle$ if $m \leq N$, where N is the length of the longest trace in T, and $r^T(m) = r^T(N)$ if $m \geq N$; that is, the final state repeats forever. Given an asynchronous trace system Σ, let $\mathcal{R}(\Sigma)$ denote the set of all runs of the form r^T, where T is a maximal set of run-like traces in Σ.

Trace-based security properties are usually expressed as closure properties on sets of traces, much like our possibilistic definitions of secrecy; see [53] for more details. We focus here on the definitions of asynchronous separability and generalized noninterference given by Zakinthinos and Lee [94].

Definition 25 *An asynchronous trace system Σ satisfies* asynchronous separability *if, for all traces $t, t' \in \Sigma$, if t'' is a trace that results from an arbitrary interleaving of the traces $t \restriction L$ and $t' \restriction H$, then $t'' \in \Sigma$.*

The definition of generalized noninterference is slightly more complicated, because the trace that results from interleaving does not include high inputs:

Definition 26 *An asynchronous trace system Σ satisfies* asynchronous generalized noninterference *if, for all traces $t, t' \in \Sigma$, if t'' is a trace that results from an arbitrary interleaving of the traces $t \restriction L$ and $t' \restriction \mathbf{In}(H)$, there exists a trace t''' such that $t''' \restriction (\mathbf{Ev}(L) \cup \mathbf{In}(H)) = t'' \restriction (\mathbf{Ev}(L) \cup \mathbf{In}(H))$.*

It is straightforward to relate these definitions to secrecy. Exactly as in the synchronous case, let f_{hi} be an information function that extracts a high input trace prefix from a point: if $r^T(m) = \langle t \restriction L, t \restriction H \rangle$, let $f_{hi}(r^T, m) = t \restriction \mathbf{In}(H)$.

Proposition 12 *If Σ is an asynchronous trace system that satisfies asynchronous separability (resp. asynchronous generalized noninterference), then H maintains total secrecy (resp. total f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

The converse of Proposition 12 does not necessarily hold. We demonstrate this by providing a counterexample that works for both separability and generalized noninterference. Suppose that there are no high output events, only one low output event l_o, and arbitrary sets $\mathbf{In}(L)$ and $\mathbf{In}(H)$ of low and high input events, respectively. Consider the system consisting of all traces t involving these events such that l_o occurs at most once in t, and when it occurs, it does not follow any high input events. In $\mathcal{R}(\Sigma)$, H maintains total secrecy and f_{hi}-secrecy with respect to L, because any local state for L is compatible with any local state for H. (Because the system is asynchronous, L learns nothing by seeing l_o: when L sees l_o, he thinks it possible that arbitrarily many high input events could have occurred *after* l_o. Furthermore, L learns nothing about H when he does *not* see l_o: it is always possible that no high input events have occurred and that l_o may yet occur.) However, Σ does not satisfy asynchronous separability or asynchronous generalized noninterference, because interleavings where a high input event precedes l_o are ruled out by construction.

This example illustrates a potential weakness of our approach to secrecy. Although H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, there is a sense in which L learns something about H. Consider a point (r, m) in $\mathcal{R}(\Sigma)$ at which L has not seen l_o. At that point, L knows that *if* a high event has occurred, he will never see l_o. This knowledge does not violate secrecy, because it does not depend on the local state of H; it is not an H-local fact. But there is a sense in which this fact can be said to be "about" H: it is information about a correlation be-

tween high events and a particular low event. However, it is unclear whether this constitutes an information flow; at any rate, we have not been able to construct an example where this knowledge is a problem. But it is worth pointing out that all of our definitions of secrecy aim to protect the local state of some particular user, and therefore that any "secret information" that cannot be characterized as a local proposition is not protected.

In any case, we can show that total secrecy and separability are equivalent if we assume a particularly strong form of asynchrony that rules out a temporal dependence between high and low events. Formally, Σ is *closed under interleavings* if for all asynchronous traces t and t', if $t \in \Sigma$, $t' \restriction L = t \restriction L$ and $t' \restriction H = t \restriction H$, then $t' \in \Sigma$. Though this requirement allows L to learn about high events that may occur in the future (or that have possibly occurred in the past), it rules out any knowledge of the ordering of high and low events in a given run. With this requirement, total secrecy and asynchronous separability coincide.

Proposition 13 *If Σ is an asynchronous trace system that is closed under interleavings, then Σ satisfies asynchronous separability iff H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

A similar result is true for generalized noninterference and f_{hi}-secrecy if we modify the definition of closure under interleavings to allow L to learn something about the ordering of high output events; we omit the details.

5.1.2 Secrecy and user strategies

Program P_1, described in Section 3.2.4, allows a high agent to transmit arbitrarily long data strings directly to a low agent even though the high agent's actual input

values remain secret. We described one possible way to model the lack of secrecy in the resulting system, by assuming that the string that high wants to transmit is included in his initial local state. This example demonstrates that generalized noninterference (as defined in the previous section) is insufficient to ensure that one agent cannot interfere with another, even if we restrict our concern to possibilistic information flows. By using a clever-enough strategy, an agent may be able to exploit the nondeterminism of a system to transmit data to another agent.

Wittbold and Johnson, who first introduced the insecure one-time pad example [93], also introduced *nondeducibility on strategies* to protect the strategy employed by high agents. We modify their definition slightly so that it is compatible with McLean's framework of synchronous traces. A *strategy* ω_H is a function from a high input/output trace prefix $t_k \restriction H$ to a high input value $h_i \in \mathbf{In}(H)$. Intuitively, a strategy tells the high agent what to do at each step, given what he has already seen and done. A trace t is *consistent* with a strategy ω_H if, for all k, $\omega_H(t_{k-1} \restriction H) = h_i^{(k)}$, where $h_i^{(k)}$ is the high input value of the kth tuple in t. A synchronous trace system Σ *satisfies nondeducibility on strategies* if, for all traces $t \in \Sigma$ and every high strategy ω_H consistent with some trace in Σ, there exists a trace t' that is consistent with ω_H such that $t' \restriction L = t \restriction L$. If the strategy of the high agent is included as part of her local state, and f_{prot} is an H-information function that extracts the strategy of the high agent from the local state, then it is straightforward to show that nondeducibility on strategies is simply synchronous f_{prot}-secrecy.

Gray and Syverson [32] extend nondeducibility on strategies to probabilistic systems using the Halpern-Tuttle framework. In Gray and Syverson's terminology, low and high agents use probabilistic strategies ω_L and ω_H, respectively. Again,

the strategies (ω_L and ω_H) determine what the agents will input next, given what they have seen and done so far. The system is assumed to have a fixed probability distribution $\mathbf{O}$ that determines its output behavior, given the inputs and outputs seen so far. Formally, for each trace prefix t of length k, $\omega_H(\cdot \,|\, (t \restriction H))$ is a probability measure on high input events that occur at time $k+1$, given the restriction of t to the high input/output; similarly, $\omega_L(\cdot \,|\, (t \restriction L))$ is a probability measure on low input events that occur at time $k+1$ and $\mathbf{O}(\cdot \,|\, t)$ is a probability measure on output events that occur at time $k+1$, given t. Gray and Syverson require that the choices made by low agent, the high agent, and the system be probabilistically independent at each time step. With this assumption, ω_H, ω_L, and $\mathbf{O}$ determine a conditional distribution that we denote $\mu_{L,H,\mathbf{O}}$, where

$$\mu_{L,H,\mathbf{O}}(\langle l_i, h_i, l_o, h_o\rangle \,|\, t) = \omega_L(l_i \,|\, (t \restriction L)) \cdot \omega_H(h_i \,|\, (t \restriction H)) \cdot \mathbf{O}(l_o, h_o \,|\, t).$$

Let Λ and Γ be countable sets of strategies for the low and high agent, respectively.[1] Given Λ, Γ, and $\mathbf{O}$ (and, implicitly, sets of low and high input and output values), we can define an adversarial probability system $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ in a straightforward way. Let Σ consist of all synchronous traces over the input and out values. For each joint strategy $(\omega_L, \omega_H) \in \Lambda \times \Gamma$, let $\mathcal{R}(\Sigma, \omega_L, \omega_H)$ consist of all runs defined as in our earlier mapping from synchronous traces to runs, except that now we include ω_L in the low agent's local state and ω_H in the high agent's local state. Let $\mathcal{R}(\Sigma, \Lambda \times \Gamma) \triangleq \cup_{(\omega_L,\omega_H)\in\Lambda\times\Gamma}\mathcal{R}(\Sigma, \omega_L, \omega_H)$. We can partition $\mathcal{R}(\Sigma, \Lambda \times \Gamma)$ according to the joint strategy used; let $\mathcal{D}(\Lambda, \Gamma)$ denote this partition. Given the independence assumptions, the joint strategy (ω_L, ω_H) also determines a probabil-

[1] Gray and Syverson take Λ and Γ to consist of *all possible* probabilistic strategies for the low and high agent, respectively, but their approach still makes sense if Λ and Γ are arbitrary sets of strategies, and it certainly seems reasonable to assume that there are only countably many strategies that agents might be using.

ity $\mu_{L,H,\mathbf{O}}$ on $\mathcal{R}(\Sigma, \omega_L, \omega_H)$. Let $\Delta(\Lambda, \Gamma, \mathbf{O}) \triangleq \{\mu_{L,H,\mathbf{O}} : \omega_L \in \Lambda, \omega_H \in \Gamma\}$, and let $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O}) \triangleq (\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{D}, \Delta\}$. We can now define Gray and Syverson's notion of secrecy in the context of these adversarial systems. (Again, L and H refer to the low agent and the high agent.)

Definition 27 *An adversarial system* $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ *satisfies* probabilistic noninterference *if, for all low strategies* $\omega_L \in \Lambda$*, points* (r, m) *where L's strategy is* ω_L*, and high strategies* $\omega_H, \omega_{H'} \in \Gamma$*, we have*

$$\mu_{(L,H,\mathbf{O})}(\mathcal{K}_L(r, m)) = \mu_{(L,H',\mathbf{O})}(\mathcal{K}_L(r, m)).$$

Theorem 6 *The following are equivalent:*

(a) $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ *satisfies probabilistic noninterference;*

(b) L obtains no evidence about H's strategy (in the sense of Definition 13) in $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$*;*

(c) H maintains generalized run-based probabilistic f_{prot}*-secrecy with respect to L in* $(\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{M}^{INIT}(\Delta(\Lambda, \Gamma, \mathbf{O})))$*, where* f_{prot} *is the information function that maps from H's local state to H's strategy;*

(d) H maintains generalized probabilistic synchronous f_{prot}*-secrecy with respect to L in the standard generalized probability system determined by* $(\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{M}^{INIT}(\Delta(\Lambda, \Gamma, \mathbf{O})))$.

Proof: The fact that (a) implies (b) is immediate from the definitions, since H's initial choice is the strategy ω_H. The equivalence of (b) and (c) follows from Theorem 5. Finally, since the traces in Σ are synchronous, the equivalence of (c) and (d) follows from Proposition 8. □

5.2 Related definitions of anonymity

5.2.1 Knowledge-based definitions of anonymity

As mentioned in the introduction, we are not the first to use knowledge to handle definitions of anonymity. In particular, Syverson and Stubblebine [89] use an epistemic logic to formalize anonymity. However, the focus of their work is very different from ours. They describe a logic for reasoning about anonymity and a number of axioms for the logic. An agent's knowledge is based, roughly speaking, on his recent actions and observations, as well as what follows from his log of system events. The first five axioms that Syverson and Stubblebine give are the standard **S5** axioms for knowledge. There are well-known soundness and completeness results relating the **S5** axiom system to Kripke structure semantics for knowledge [20]. However, they give many more axioms, and they do not attempt to give a semantics for which their axioms are sound. Our focus, on the other hand, is completely semantic. We have not tried to axiomatize anonymity. Rather, we try to give an appropriate semantic framework in which to consider anonymity.

In some ways, Syverson and Stubblebine's model is more detailed than the model used here. Their logic includes many formulas that represent various actions and facts, including the sending and receiving of messages, details of encryption and keys, and so on. They also make more assumptions about the local state of a given agent, including details about the sequence of actions that the agent has performed locally, a log of system events that have been recorded, and a set of facts of which the agent is aware. While these extra details may accurately reflect the nature of agents in real-world systems, they are orthogonal to our concerns here. In any case, it would be easy to add such expressiveness to our model as

well, simply by including these details in the local states of the various agents.

It is straightforward to relate our definitions to those of Syverson and Stubblebine. They consider facts of the form $\phi(i)$, where i is a principal, i.e., an agent. They assume that the fact $\phi(i)$ is a single formula in which a single agent name occurs. Clearly, $\theta(i, a)$ is an example of such a formula. In fact, Syverson and Stubblebine assume that if $\phi(i)$ and $\phi(j)$ are both true, then $i = j$. For the $\theta(i, a)$ formulas, this means that $\theta(i, a)$ and $\theta(i', a)$ cannot be simultaneously true: at most one agent can perform an action in a given run, exactly as in the setup of Proposition 9.

There is one definition in [89] that is especially relevant to our discussion; the other relevant definitions presented there are similar. A system is said to satisfy $(\geq k)$-*anonymity* with respect to observer o if the following formula is valid:

$$\phi(i) \Rightarrow \exists i_1, \ldots, i_{k-1} \,.\, P_o(\phi(i)) \wedge P_o(\phi(i_1)) \wedge \cdots \wedge P_o(\phi(i_{k-1})).$$

This definition says that if $\phi(i)$ holds, there must be at least k agents, including i, that the observer suspects. The definition is essentially equivalent to our definition of $(k-1)$-anonymity. If $P_o(\phi(i'))$ is true for $k-1$ agents other than i, then the formula must hold, because $\phi(i) \Rightarrow P_o(\phi(i))$ is valid.

5.2.2 CSP and anonymity

A great deal of work on the foundations of computer security has used process algebras such as CCS and CSP [62, 46] as the basic system framework [22, 78]. Process algebras offer several advantages: they are simple, they can be used for specifying systems as well as system properties, and model-checkers are available that can be used to verify properties of systems described using their formalisms.

Schneider and Sidiropoulos [79] use CSP both to characterize one type of anonymity and to describe variants of the dining cryptographers problem [5]. They then use a model-checker to verify that their notion of anonymity holds for those variants of the problem. To describe their approach, we need to outline some of the basic notation and semantics of CSP. To save space, we give a simplified treatment of CSP here. (See Hoare [46] for a complete description of CSP.) The basic unit of CSP is the *event*. Systems are modeled in terms of the events that they can perform. Events may be built up several components. For example, "donate.$5" might represent a "donate" event in the amount of $5. *Processes* are the systems, or components of systems, that are described using CSP. As a process unfolds or executes, various events occur. For our purposes, we make the simplifying assumption that a process is determined by the event sequences it is able to engage in.

As with some of the previous system models considered, we can associate with every process a set of traces. Intuitively, each trace in the set associated with process P represents one sequence of events that might occur during an execution of P. Informally, CSP event traces correspond to finite prefixes of runs, except that they do not explicitly describe the local states of agents and do not explicitly describe time.

Schneider and Sidiropoulos define a notion of anonymity with respect to a set A of events. Typically, A consists of events of the form $i.a$ for a fixed action a, where i is an agent in some set that we denote I_A. Intuitively, anonymity with respect to A means that if any event in A occurs, it could equally well have been any other event in A. In particular, this means that if an agent in I_A performs a, it could equally well have been any other agent in I_A. Formally, given a set Σ of possible

events and $A \subseteq \Sigma$, let f_A be a function on traces that, given a trace t, returns a trace $f_A(t)$ that is identical to t except that every event in A is replaced by a fixed event $\alpha \notin \Sigma$. A process P is *strongly anonymous* on A if $f_A^{-1}(f_A(P)) = P$, where we identify P with its associated set of traces. This means that all the events in A are interchangeable; by replacing any event in A with any other we would still get a valid trace of P.

Schneider and Sidiropoulos give several very simple examples that are useful for clarifying this definition of anonymity. One is a system where there are two agents who can provide donations to a charity, but where only one of them will actually do so. Agent 0, if she gives a donation, gives \$5, and agent 1 gives \$10. This is followed by a "thanks" from the charity. The events of interest are "0.gives" and "1.gives" (representing events where 0 and 1 make a donation), "\$5" and "\$10" (representing the charity's receipt of the donation), "thanks," and "STOP" (to signify that the process has ended). There are two possible traces:

1. 0.gives → \$5 → "thanks" → STOP.

2. 1.gives → \$10 → "thanks" → STOP.

The donors require anonymity, and so we require that the CSP process is strongly anonymous on the set {0.gives, 1.gives}. In fact, this condition is not satisfied by the process, because "0.gives" and "1.gives" are not interchangeable. This is because "0.gives" must be followed by "\$5," while "1.gives" must be followed by "\$10." Intuitively, an agent who observes the traces can determine the donor by looking at the amount of money donated.

We believe that Schneider and Sidiropoulos's definition is best understood as trying to capture the intuition that an observer who sees all the events generated

by P, except for events in A, does not know which event in A occurred. We can make this precise by translating Schneider and Sidiropoulos's definition into our framework. The first step is to associate with each process P a corresponding set of runs $\mathcal{R}_P$. We present one reasonable way of doing so here, which suffices for our purposes. In future work, we hope to explore the connection between CSP and the runs and systems framework in more detail.

Recall that a run is an infinite sequence of global states of the form

$$(s_e, s_1, \ldots, s_n),$$

where each s_i is the local state of agent i, and s_e is the state of the environment. Therefore, to specify a set of runs, we need to describe the set of agents, and then explain how to derive the local states of each agent for each run. There is an obvious problem here: CSP has no analogue of agents and local states. To get around this, we could simply tag all events with an agent (as Schneider and Sidiropoulos in fact do for the events in A). However, for our current purposes, a much simpler approach will do. The only agent we care about is a (possibly mythical) observer who is able to observe every event except the ones in A. Moreover, for events in A, the observer knows that something happened (although not what). There may be other agents in the system, but their local states are irrelevant. We formalize this as follows.

Fix a process P over some set Σ of events, and let $A \subseteq \Sigma$. Following Schneider and Sidiropoulos, for the purposes of this discussion, assume that A consists of events of the form $i.a$, where $i \in I_A$ and a is some specific action. We say that a system $\mathcal{R}$ is *compatible with* P if there exists some agent o such that the following two conditions hold:

- for every run $r \in \mathcal{R}$ and every time m, there exists a trace $t \in P$ such that

$t = r_e(m)$ and $f_A(t) = r_o(m)$;

- for every trace $t \in P$, there exists a run $r \in \mathcal{R}$ such that $r_e(|t|) = t$ and $r_o(|t|) = f_A(t)$ (where $|t|$ is the number of events in t).

Intuitively, $\mathcal{R}$ represents P if (1) for every trace t in P, there is a point (r, m) in $\mathcal{R}$ such that, at this point, exactly the events in t have occurred (and are recorded in the environment's state) and o has observed $f_A(t)$, and (2) for every point (r, m) in $\mathcal{R}$, there is a trace t in P such that precisely the events in $r_e(m)$ have happened in t, and o has observed $f_A(t)$ at (r, m). We say that the interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is *compatible with* P if $\mathcal{R}$ is compatible with P and if $(\mathcal{I}, r, m) \models \theta(i, a)$ whenever the event $i.a$ is in the event sequence $r_e(m')$ for some m'.

We are now able to make a formal connection between our definition of anonymity and that of Schneider and Sidiropoulos. As in the setup of Proposition 9, we assume that an anonymous action a can be performed only once in a given run.

Theorem 7 *If $\mathcal{I} = (\mathcal{R}, \pi)$ is compatible with P, then P is strongly anonymous on the alphabet A if and only if for every agent $i \in I_A$, the action a performed by i is anonymous up to I_A with respect to o in $\mathcal{I}$.*

Up to now, we have assumed that the observer o has access to all the information in the system except which event in A was performed. Schneider and Sidiropoulos extend their definition of strong anonymity to deal with agents that have somewhat less information. They capture "less information" using *abstraction operators*. Given a process P, there are several abstraction operators that can give us a new process. For example the *hiding operator*, represented by $\backslash$, hides all events in some set C. That is, the process $P\backslash C$ is the same as P except that all events in C become internal events of the new process, and are not included

in the traces associated with $P\backslash C$. Another abstraction operator, the renaming operator, has already appeared in the definition of strong anonymity: for any set C of events, we can consider the function f_C that maps events in C to a fixed new event. The difference between hiding and renaming is that, if events in C are hidden, the observer is not even aware they took place. If events in C are renamed, then the observer is aware that some event in C took place, but does not know which one.

Abstraction operators such as these provide a useful way to model a process or agent who has a distorted or limited view of the system. In the context of anonymity, they allow anonymity to hold with respect to an observer with a limited view of the system in cases where it would not hold with respect to an observer who can see everything. In the anonymous donations example, hiding the events \$5 and \$10, i.e., the amount of money donated, would make the new process $P\backslash\{\$5, \$10\}$ strongly anonymous on the set of donation events. Formally, given an abstraction operator ABS_C on a set of events C, we have to check the requirement of strong anonymity on the process $ABS_C(P)$ rather than on the process P.

Abstraction is easily captured in our framework. It amounts simply to changing the local state of the observer. For example, anonymity of the process $P\backslash C$ in our framework corresponds to anonymity of the action a for every agent in I_A with respect to an observer whose local state at the point (r, m) is $f_A(r_e(m))\backslash C$. We omit the obvious analogue of Theorem 7 here.

A major advantage of the runs and systems framework is that definitions of high-level properties such as anonymity do not depend on the local states of the agents in question. If we want to model the fact that an observer has a limited view of the system, we need only modify her local state to reflect this fact. While

some limited views are naturally captured by CSP abstraction operators, others may not be. The definition of anonymity should not depend on the existence of an appropriate abstraction operator able to capture the limitations of a particular observer.

As we have demonstrated, our approach to anonymity is compatible with the approach taken in [79]. Our definitions are stated in terms of actions, agents, and knowledge, and are thus very intuitive and flexible. The generality of runs and systems allows us to have simple definitions that apply to a wide variety of systems and agents. The low-level CSP definitions, on the other hand, are more operational than ours, and this allows easier model-checking and verification. Furthermore, there are many advantages to using process algebras in general: systems can often be represented much more succinctly, and so on. This suggests that both approaches have their advantages. Because CSP systems can be represented in the runs and systems framework, however, it makes perfect sense to define anonymity for CSP processes using the knowledge-based definitions we have presented here. If our definitions turn out to be equivalent to more low-level CSP definitions, this is ideal, because CSP model-checking programs can then be used for verification. A system designer simply needs to take care that the runs-based system derived from a CSP process (or set of processes) represents the local states of the different agents appropriately.

5.2.3 Anonymity and function-view semantics

Hughes and Shmatikov [47] introduce *function views* and function-view *opaqueness* as a way of expressing a variety of confidentiality properties in a succinct and uniform way. Their main insight is that requirements such as anonymity involve

restrictions on relationships between entities such as agents and actions. Because these relationships can be expressed by functions from one set of entities to another, hiding information from an observer amounts to limiting an observer's view of the function in question. For example, anonymity properties are concerned with whether or not an observer is able to connect actions with the agents who performed them. By considering the function from the set of actions to the set of agents who performed those actions, and specifying the degree to which that function must be opaque to observers, we can express anonymity using the function-view approach.

To model the uncertainty associated with a given function, Hughes and Shmatikov define a notion of *function knowledge* to explicitly represent an observer's partial knowledge of a function. Function knowledge focuses on three particular aspects of a function: its graph, image, and kernel. (Recall that the *kernel* of a function f with domain X is the equivalence relation ker on X defined by $(x, x') \in ker$ iff $f(x) = f(x')$.) *Function knowledge* of type $X \rightarrow Y$ is a triple $N = (F, I, K)$, where $F \subseteq X \times Y$, $I \subseteq Y$, and K is an equivalence relation on X. A triple (F, I, K) is *consistent with* f if $f \subseteq F$, $I \subseteq imf$, and $K \subseteq kerf$. Intuitively, a triple (F, I, K) that is consistent with f represents what an agent might know about the function f. Complete knowledge of a function f, for example, would be represented by the triple $(f, imf, kerf)$.

With respect to anonymity (and other confidentiality properties), we are interested not in what an agent knows, but in what an agent does *not* know. This is formalized by Hughes and Shmatikov in terms of opaqueness conditions for function knowledge. If $N = \langle F, I, K \rangle$ is consistent with $f : X \rightarrow Y$, then, for example, N is *k-value opaque* if $|F(x)| \geq k$ for all $x \in X$. That is, N is k-value opaque if there are k possible candidates for the value of $f(x)$, for all $x \in X$. Similarly, N

is *Z-value opaque* if $Z \subseteq F(x)$ for all $x \in X$. In other words, for each x in the domain of f, no element of Z can be ruled out as a candidate for $f(x)$. Finally, N is *absolutely value opaque* if that N is Y-value opaque.

Opaqueness conditions are closely related to the nonprobabilistic definitions of anonymity given in Section 4.1. Consider functions from X to Y, where X is a set of actions and Y is a set of agents, and suppose that some function f is the function that, given some action, names the agent who performed the action. If we have k-value opaqueness for some view of f (corresponding to some observer o), this means, essentially, that each action a in X is k-anonymous with respect to o. Similarly, the view is I_A-value opaque if the action is anonymous up to I_A for each agent $i \in I_A$. Thus, function view opaqueness provides a concise way of describing anonymity properties.

To make these connections precise, we need to explain how function views can be embedded within the runs and systems framework. Hughes and Shmatikov already show how we can define function views using Kripke structures, the standard approach for giving semantics to knowledge. A minor modification of their approach works in systems too. Assume we are interested in who performs an action $a \in X$, where X, intuitively, is a set of "anonymous actions." Let Y be the set of agents, including a "nobody agent" denoted N, and let f be a function from X to Y. Intuitively, $f(a) = i$ if agent i performs action a, and $f(a) = N$ if no agent performs action a. The value of the function f will depend on the point. Let $f_{r,m}$ be the value of f at the point (r, m). Thus, $f_{r,m}(a) = i$ if i performs a in run r. [2] We can now easily talk about function opaqueness with respect to an observer o. For example, f is Z-value opaque at the point (r, m) with respect to o if, for all

[2]Note that for $f_{(r,m)}$ to be well-defined, it must be the case that only one agent can ever perform a single action.

$z \in Z$, there exists a point (r', m') such that $r'_o(m') = r_o(m)$ and $f_{(r',m')}(x) = z$. In terms of knowledge, Z-value opaqueness says that for any value x in the range of f, o thinks it possible that any value $z \in Z$ could be the result of $f(x)$. Indeed, Hughes and Shmatikov say that function-view opaqueness, defined in terms of Kripke structure semantics, is closely related to epistemic logic. The following proposition makes this precise; it would be easy to state similar propositions for other kinds of function-view opaqueness.

Proposition 14 *Let $\mathcal{I} = (\mathcal{R}, \pi)$ be an interpreted system that satisfies $(\mathcal{I}, r, m) \models f(x) = y$ whenever $f_{(r,m)}(x) = y$. In system $\mathcal{I}$, f is Z-value opaque for observer o at the point (r, m) if and only if*

$$(\mathcal{I}, r, m) \models \bigwedge_{x \in X} \bigwedge_{z \in Z} P_o[f(x) = z].$$

Proof: This result follows immediately from the definitions. □

Stated in terms of knowledge, function-view opaqueness already looks a lot like our definitions of anonymity. Given f (or, more precisely, the set $\{f_{(r,m)}\}$ of functions) mapping actions to agents, we can state a theorem connecting anonymity to function-view opaqueness. There are two minor issues to deal with, though. First, our definitions of anonymity are stated with respect to a single action a, while the function f deals with a *set* of actions. We can deal with this by taking the domain of f to be the singleton $\{a\}$. Second, our definition of anonymity up to a set I_A requires the observer to suspect agents in I_A only if i actually performs the action a. (Recall this is also true for Syverson and Stubblebine's definitions.) I_A-value opaqueness requires the observer to think many agents could have performed an action even if nobody has. To deal with this, we require opaqueness only when the action has been performed by one of the agents in I_A.

Theorem 8 *Suppose that* $(\mathcal{I}, r, m) \models \theta(i, a)$ *exactly if* $f_{(r,m)}(a) = i$. *Then action* a *is anonymous up to* I_A *with respect to* o *for each agent* $i \in I_A$ *if and only if at all points* (r, m) *such that* $f_{(r,m)}(a) \in I_A$, f *is* I_A*-value opaque with respect to* o.

As with Proposition 14, it would be easy to state analogous theorems connecting our other definitions of anonymity, including minimal anonymity, total anonymity, and k-anonymity, to other forms of function-view opaqueness. We omit the details here.

The assumptions needed to prove Theorem 8 illustrate two ways in which our approach may seem to be less general than the function-view approach. First, all our definitions are given with respect to a single action, rather than with respect to a set of actions. However, it is perfectly reasonable to specify that all actions in some set $\mathcal{A}$ of actions be anonymous. Then we could modify Theorem 8 so that the function f is defined on all actions in $\mathcal{A}$. (We omit the details.) Second, our definitions of anonymity only restrict the observer's knowledge if somebody actually performs the action. This is simply a different way of defining anonymity. As mentioned previously, we are not trying to give a definitive definition of anonymity, and it certainly seems reasonable that someone might want to define or specify anonymity using the stronger condition. At any rate, it would be straightforward to modify our definitions so that the implications, involving $\theta(i, a)$, are not included.

Hughes and Shmatikov argue that epistemic logic is a useful language for expressing anonymity specifications, while CSP is a useful language for describing and specifying systems. We agree with both of these claims. They propose function views as a useful interface to mediate between the two. We have tried to argue here that no mediation is necessary, since the multiagent systems framework can

also be used for describing systems. (Indeed, the traces of CSP can essentially be viewed as runs.) Nevertheless, we do believe that function views can be the basis of a useful language for reasoning about some aspects of confidentiality. We can well imagine adding abbreviations to the language that let us talk directly about function views. (We remark that we view these abbreviations as syntactic sugar, since these are notions that can already be expressed directly in terms of the knowledge operators we have introduced.)

On the other hand, we believe that function views are insufficiently expressive. One obvious problem is adding probability. While it is easy to add probability to systems, as we have shown, and to capture interesting probabilistic notions of anonymity, it is far from clear how to do this if we take function views triples as primitive.

To sum up, we would argue that to reason about knowledge and probability, we need to have possible worlds as the underlying semantic framework. Using the multiagent systems approach gives us possible worlds in a way that makes it particularly easy to relate them to systems. Within this semantic framework, function views may provide a useful syntactic construct with which to reason about anonymity and other confidentiality properties.

Chapter 6

Information-Flow Security for Interactive Programs

In this chapter we consider secrecy in systems described by imperative programs. As mentioned in the Introduction, we are concerned with interactive programs, which interact with users at runtime, rather than batch-job programs that can be characterized as a function from inputs to outputs. Our language and security conditions synthesize two branches of information-flow security research, in that we leverage the trace-based definitions that have been proposed for interactive systems to provide novel security conditions for imperative programs. Furthermore, our interactive programming language can be viewed as a specification language for interactive systems that more closely approximates the implementation of real programs than the abstract system models that have previously been used. Our goal is to leverage the definitions of secrecy given in Chapter 3 to state noninterference requirements for interactive systems directly in terms of the operational semantics of an imperative language.

In giving definitions of information-flow security for interactive programs, we pay special attention to the issue of nondeterminism, which arises in real-world systems for a number of reasons (including concurrency and probabilistic randomization). Nondeterminism is orthogonal to interactivity, but the interplay between information flow and nondeterminism is often quite subtle. We examine two kinds of nondeterministic choices: those which we assume are made probabilistically, and those which we are unable or unwilling to assign probabilities. We refer to the former as *probabilistic choice*, and to the latter as *nondeterministic choice*. Following

the approach taken in Chapter 3, we factor out nondeterministic choice so that we can reason about it in isolation from probabilistic choice. By explicitly representing the resolution of nondeterministic choice in the language semantics, we adapt our security condition to rule out *refinement attacks* in which the resolution of nondeterministic choice results in insecure information flows. (We thus explicitly rule out possibilistic breaches of secrecy, as discussed in earlier chapters.) We also give a probabilistic security condition based on Definition 13 in Section 3.2.5.

6.1 User strategies

It might seem at first that information-flow security for interactive programs can be obtained by adopting the same approach used for batch-job programs, that is, by preventing low users from learning anything about high inputs. However, several papers, starting with Wittbold and Johnson [93], have described systems in which high users can transmit information to low users even though low users learn nothing about the high inputs. This is demonstrated by Program P_1, which we described in Section 3.2.4. We reproduce P_1 here. Recall that **input** x **from** C reads a value from a channel named C and stores it in variable x; similarly, **output** e **to** C outputs the value of expression e on a channel named C. Assume that low users may use only channel L, that high users may use channel H, and that no users may observe the values of program variables. Infix operator ${}_p[]$ executes its first operand with probability p and its second operand with probability $1 - p$.

P_1 : **while** (**true**) **do**
  $x := 0 \;{}_{0.5}[]\; x := 1;$
  output x **to** H;
  input y **from** H;
  output x **xor** $(y$ **mod** $2)$ **to** L

Because the probabilistic choice is resolved in a way that is unpredictable to the low user, he will be unable to determine the inputs on channel H: for any output on L, the input on H could have been either 0 or 1. Yet the high user can still communicate an arbitrary confidential bit z to channel L at each iteration of the loop by choosing z **xor** x as input on H.

The confidential information z is never directly acquired by the program: it is neither the initial value of a program variable nor an input supplied on a channel. As Wittbold and Johnson observe, maintaining the secrecy of all high inputs (and even the initial values of program variables) is therefore insufficient to preserve the secrecy of confidential information.

In Program P_1, the high user is able to communicate arbitrary confidential information by selecting his next input as a function of outputs he has previously received. This suggests that if we want to prevent confidential information from flowing to low users, we should protect the secrecy of the function that high users employ to select inputs. Following Wittbold and Johnson's terminology, we call this function a *user strategy*. In the remainder of this section we develop the mathematical structures needed to define user strategies formally.

6.1.1 Types, users, and channels

We assume a set $\mathcal{L}$ of security types with ordering relation $\leq$ and use metavariable τ to range over security types. For simplicity, we assume that $\mathcal{L}$ equals $\{L, H\}$ with $L \leq H$. (Our results generalize to partial orders of security types.) Security type L represents low confidentiality, and H represents high confidentiality. The ordering $\leq$ indicates the relative restrictiveness of security types: high-confidentiality information is more restricted in its use than low-confidentiality information.

Users are agents (including humans and programs) that interact with executing programs. We associate with each user a security type indicating the highest level of confidential information that the user is permitted to read. Conservatively, we assume that users of the same security type may collaborate while attempting to subvert the security of a program. We can thus simplify our security analyses by reasoning about exactly two users, one representing the pooled knowledge of low users and another representing the pooled knowledge of high users.

We also assume the existence of *channels* with blocking input and nonblocking output. Although input is blocking, we assume that all inputs prompted for are eventually supplied. Each channel is associated with a security type τ, and only users of that type are permitted to use the channel. For simplicity, we assume that there are exactly two channels, L and H. We also assume that the values that are input and output on channels are integers. These are not fundamental restrictions; our results could be extended to allow multiple channels of each type, to allow high users to observe low channels, and to allow more general data types.

6.1.2 Traces

An *event* is the transmission of an input or output on a channel. Denote the input of value v on the channel of type τ as $in(\tau, v)$ and the output of v on τ as $out(\tau, v)$. Let $\mathbf{Ev}(\tau)$ be the set of all events that could occur on channel τ:

$$\mathbf{Ev}(\tau) \quad \triangleq \quad \bigcup_{v \in \mathbb{Z}} \{in(\tau, v), out(\tau, v)\}.$$

Let $\mathbf{Ev}$ be the set of all events:

$$\mathbf{Ev} \quad \triangleq \quad \bigcup_{\tau \in \mathcal{L}} \mathbf{Ev}(\tau).$$

We use metavariable α to range over events in $\mathbf{Ev}$.

A *trace* is a finite list of events. Given $E \subseteq \mathbf{Ev}$, an *event trace on* E is a finite, possibly empty list $\langle \alpha_1, \ldots, \alpha_n \rangle$ such that $\alpha_i \in E$ for all i. The empty trace is written $\langle\rangle$. The set of all traces on E is denoted $\mathbf{Tr}(E)$, and we abbreviate the set of all traces $\mathbf{Tr}(\mathbf{Ev})$ as $\mathbf{Tr}$. Trace equality is defined pointwise, and the concatenation of two traces t and t' is denoted $t \hat{} t'$. A trace t' *extends* trace t if there exists a trace t'' such that $t' = t \hat{} t''$. The *restriction of* t *to* E, denoted $t \restriction E$, is the trace that results from removing all events not contained in E from t. We write $t \restriction \tau$ as shorthand for $t \restriction \mathbf{Ev}(\tau)$. A *low trace* is the low restriction $t \restriction L$ of a trace t.

6.1.3 User strategies

As demonstrated by Program P_1, the input supplied by a user may depend on past events observed by that user. To capture this dependence we employ a *user strategy*, which determines the input for a particular channel as a function of the events that occur on the channel. Because events on a channel include both inputs and outputs, this function depends on both the user's observations and previous actions. Formally, a user strategy for a channel with security type τ is a function of type $\mathbf{Tr}(\mathbf{Ev}(\tau)) \to \mathbb{Z}$. Let **UserStrat** be the set of all user strategies. (Note that, to simulate the batch-job model, the initial inputs provided by users can be represented by a constant strategy that selects inputs without regard for past inputs or outputs. Also, high user strategies can be extended to depend on observation of the low channel, as described at the end of Section 6.2.)

As an example, we present a strategy that a high user could employ to transmit an arbitrary stream of bits $z_1 z_2 \ldots$ to the low user in Program P_1. This user strategy, g, ensures that if b was the previous output on H, then the next input on

H is the bitwise exclusive-or of b and z_i. Note that every second event on channel H is an input event $in(H, v)$.

$$g(\langle \alpha_1, \ldots, \alpha_n \rangle) = \begin{cases} z_i \text{ xor } b & \text{if } \alpha_n = out(H, b) \text{ and } n = 2i - 1 \\ 0 & \text{otherwise} \end{cases}$$

A *joint strategy* is a collection of user strategies, one for each channel. Formally, a joint strategy ω is a function of type $\mathcal{L} \to \mathbf{UserStrat}$, that is, a function from security types to user strategies. Let **Strat** be the set of all joint strategies.

6.2 Noninterference for interactive programs

While-programs, extended with commands for input and output, constitute our core interactive programming language. The syntax of this language is:

$$\begin{array}{lll} \text{(expressions)} & e ::= n \mid x \mid e_0 \oplus e_1 \\ \text{(commands)} & c ::= \textsf{skip} \mid x := e \mid c_0; c_1 \mid \\ & \qquad \textsf{input } x \textsf{ from } \tau \mid \textsf{output } e \textsf{ to } \tau \mid \\ & \qquad \textsf{if } e \textsf{ then } c_0 \textsf{ else } c_1 \mid \textsf{while } e \textsf{ do } c \end{array}$$

Metavariable x ranges over **Var**, the set of all program variables. Variables take values in $\mathbb{Z}$, the set of integers. Literal values n also range over integers. Binary operator $\oplus$ denotes any total binary operation on the integers.

6.2.1 Operational semantics

The execution of a program modifies the values of variables and produces events on channels. A *state* determines the values of variables. Formally, a state is a function of type **Var** $\to$ $\mathbb{Z}$. Let σ range over states. A *configuration* is a 4-tuple (c, σ, t, ω) representing a system about to execute c with state σ and joint strategy

(Assign)

$$\frac{}{(x := e, \sigma, t, \omega) \longrightarrow (\textsf{skip}, \sigma[x := \sigma(e)], t, \omega)}$$

(Seq-1)

$$\frac{}{(\textsf{skip}; c, \sigma, t, \omega) \longrightarrow (c, \sigma, t, \omega)}$$

(Seq-2)

$$\frac{(c_0, \sigma, t, \omega) \longrightarrow (c_0', \sigma', t', \omega)}{(c_0; c_1, \sigma, t, \omega) \longrightarrow (c_0'; c_1, \sigma', t', \omega)}$$

(In)

$$\frac{\omega(\tau)(t \restriction \tau) = v}{(\textsf{input } x \textsf{ from } \tau, \sigma, t, \omega) \longrightarrow (\textsf{skip}, \sigma[x := v], t \hat{} \langle in(\tau, v) \rangle, \omega)}$$

(Out)

$$\frac{\sigma(e) = v}{(\textsf{output } e \textsf{ to } \tau, \sigma, t, \omega) \longrightarrow (\textsf{skip}, \sigma, t \hat{} \langle out(\tau, v) \rangle, \omega)}$$

(If-1)

$$\frac{\sigma(e) \neq 0}{(\textsf{if } e \textsf{ then } c_0 \textsf{ else } c_1, \sigma, t, \omega) \longrightarrow (c_0, \sigma, t, \omega)}$$

(If-2)

$$\frac{\sigma(e) = 0}{(\textsf{if } e \textsf{ then } c_0 \textsf{ else } c_1, \sigma, t, \omega) \longrightarrow (c_1, \sigma, t, \omega)}$$

(While)

$$\frac{}{(\textsf{while } e \textsf{ do } c, \sigma, t, \omega) \longrightarrow (\textsf{if } e \textsf{ then } (c; \textsf{while } e \textsf{ do } c) \textsf{ else skip}, \sigma, t, \omega)}$$

Figure 6.1: Operational semantics.

ω. Trace t is the history of events produced by the system so far. Let m range over configurations. Terminal configurations, which have no commands remaining to execute, have the form $(\textsf{skip}, \sigma, t, \omega)$.

The operational semantics for our language is a small-step relation $\longrightarrow$ on configurations. Membership in the relation is denoted

$$(c, \sigma, t, \omega) \longrightarrow (c', \sigma', t', \omega),$$

meaning that execution of command c can take a single step to command c', while updating the state from σ to σ'. Trace t' extends t with any events that were produced during the step. Note that joint strategy ω is unchanged when a configuration takes a step; we include it in the configuration only to simplify notation and presentation.

The inductive rules defining relation $\longrightarrow$ are given in Figure 6.1. The rules for commands other than input and output are all standard. In Rule Assign, $\sigma(e)$ denotes the value of expression e in state σ, and state update $\sigma[x := v]$ changes the value of variable x to v in σ. Rule In uses the joint strategy ω to determine the next input event and appends it to the current trace, and rule Out simply appends the output event to the current trace.

Let $\longrightarrow^*$ be the reflexive transitive closure of $\longrightarrow$. Intuitively, if

$$(c, \sigma, t, \omega) \longrightarrow^* (c', \sigma', t', \omega),$$

then configuration (c, σ, t, ω) can reach configuration $(c', \sigma', t', \omega)$ in zero or more steps. Configuration m *emits* t, denoted $m \rightsquigarrow t$, when there exists a configuration (c, σ, t, ω) such that $m \longrightarrow^* (c, \sigma, t, \omega)$. Note that emitted events may include both inputs and outputs.

6.2.2 A strategy-based security condition

We now develop a security condition which ensures that users with access only to channel L do not learn anything about the strategies employed by users interacting with channel H. Since strategies encode the possible actions that users may take as they interact with the system, protecting the secrecy of high strategies ensures that the actions taken by high users cannot affect (or "interfere with") the observations of low users. As we will demonstrate in Section 6.5, the security condition can be seen as an instance of our definitions of secrecy.

Informally, a program is secure if, for every initial state σ, any trace of events seen on channel L is consistent with every possible user strategy for channel H. This ensures that low users cannot learn any information, including inputs, that high users attempt to convey—even if low users know the program text.

Definition 28 (Noninterference) *A command c satisfies* noninterference *if for all $m = (c, \sigma, \langle\rangle, \omega)$ and $m' = (c, \sigma, \langle\rangle, \omega')$ such that $\omega(L) = \omega'(L)$ and for all traces t such that $m \rightsquigarrow t$, there exists a t' such that $t \restriction L = t' \restriction L$ and $m' \rightsquigarrow t'$.*

According to this condition, the high strategy $\omega(H)$ in m can be replaced by any other high strategy without affecting the low traces emitted. Although the condition assumes that programs begin with an empty trace of prior events, it can be generalized to account for arbitrary traces. (See Appendix C.) Some additional implications of this security condition are discussed below.

Initial variable values. The security condition does not protect the secrecy of the initial values of variables. More concretely, the program **output** x **to** L is considered secure for any $x \in \mathbf{Var}$, whereas the program **input** x **from** H; **output** x **to** L is obviously considered insecure. The definition thus reflects our intuition that

high users interact with the system only via input and output events on the high channel and have no control over the initialization of variables. Systems in which the high user controls the initial values of some or all variables can be modeled by prepending commands that read inputs from the high user and assign them to variables.

Variable typings. It is not necessary to assign security types to program variables in order to determine whether a program is secure. (A program with no high inputs, for example, is secure regardless of its variables or their types.) Accordingly, our security condition makes no reference to the security types of variables. This distinguishes our work from most batch-job conditions, where variable typings are fundamental. We do, however, employ variable typings for the static analysis technique presented in Section 6.6.

Timing sensitivity. Our observational model is *asynchronous*: users do not observe the time when events occur or the time that passes while a program is blocking on an input command. The security condition is thus timing-insensitive. We could incorporate timing sensitivity into the model by assuming that users observe a "tick" event at each execution step or by tagging events with the time at which they occur; strategies could then make use of this additional temporal information.

Termination sensitivity. We make the standard assumption that users are unable to observe the nontermination of a program. Nonetheless, our security condition is termination-sensitive when low events follow commands that may not

terminate. Consider the following program:

$$P_2: \quad \textsf{input } x \textsf{ from } H;\ \textsf{if } (x = 0) \textsf{ then } \{\textsf{while } (\textsf{true}) \textsf{ do skip}\} \textsf{ else skip};\ \textsf{output } 1 \textsf{ to } L$$

A high user can cause this program to transmit the value 1 to a low user. Since this would allow the low user to infer something about the high strategy, this program is insecure according to our security condition.

We do not assume that users are able to observe the termination of a program directly, but it would be easy to make termination observable by adding a distinguished termination event that is broadcast on all channels when execution reaches a terminal configuration.

Observation of channels. We have assumed that high users cannot observe the low channel, but this restriction can be removed in several ways. For example, it is straightforward to amend the operational semantics to echo low events to high channels by adding an additional high output event (prepended with a label to distinguish it from a regular high output events) to the trace every time a low input or output event occurs.

6.3 Nondeterministic programs

We distinguish two kinds of nondeterminism that appear in programs: *probabilistic choice* and *nondeterministic choice*. Intuitively, probabilistic choice represents explicit use of randomization, whereas nondeterministic choice represents program behavior that is underspecified (perhaps due to unpredictable factors such as the scheduler in a concurrent setting). Following the approach of previous work [44, 90], we factor out the latter kind of nondeterminism by assuming that all nondeterministic choices are made as if they were specified before the program began execution.

(The implications of this approach are discussed at the end of the section.) This allows reasoning about nondeterministic choice and probabilistic choice in isolation, and our definitions of noninterference reflect the resulting separation of concerns. In this section we extend our model to include nondeterministic choice. We return to probabilistic choice in Section 6.4.

6.3.1 Refiners

We extend the language of Section 6.2 with nondeterministic choice:

$$c \ ::= \ \ldots \ | \ c_0 \ []_\tau \ c_1$$

Each nondeterministic choice is annotated with a security type τ that is used in the operational semantics. The need for the annotation is explained below; we remark, however, that the type system described in Section 6.6 could be used to infer annotations automatically, so that programmers need not specify them.

To factor out the resolution of nondeterminism, we introduce infinite lists of binary values called *refinement lists*. Denote the set of all such refinement lists as **RefList**. Informally, when a nondeterministic choice is encountered during execution, the head element of a refinement list is removed and used to resolve the choice. The program executes the left command of the nondeterministic choice if the element is 0 and the right command if the element is 1. Refinement lists are an operational analog of Milner's *oracle domains* [61] for denotational semantics.

We do not want choices made in accordance with refinement lists to cause unwanted information flows. More generally, nondeterministic choices should not cause insecure information flows, even if low users can predict how the choices will be made. While it might seem that using a single refinement list would suffice

to ensure that no insecure information flows arise as a result of the resolution of nondeterministic choice, the following program demonstrates that this is not the case:

input x **from** H;
if $(x = 0)$ **then** $\{$**skip** $[]_H$ **skip**$\}$ **else skip**;
output 0 **to** L $[]_L$ **output** 1 **to** L

If the refinement list $\langle 1, 0, \ldots \rangle$ is used to execute this program, the output on channel L will equal the input on channel H. An insecure information flow arises because the same refinement list is used to make both low and high choices. To eliminate this flow, we identify the security type of a choice based on its annotation and require that different lists be used to resolve choices at each type. This ensures that the number of choices made at a given security level cannot become a covert channel. (Note that this requirement lends itself to natural implementation techniques. For example, if choices are made by using a stream of pseudorandom numbers, then different streams should be used to resolve high and low choices. Or if $[]$ represents scheduler choices, then the scheduler should resolve choices at each security type independently.)

A *refiner* is a function $\psi : \mathcal{L} \to \mathbf{RefList}$ that associates a refinement list with each security type. Let **Ref** denote the set of all refiners. Denote the standard list operations of reading the first element of a list and removing the first element of a list as *head* and *tail*, respectively. Given a refiner ψ, the value $head(\psi(\tau))$ is used to resolve the next choice annotated with type τ.

6.3.2 Operational semantics

Using refiners, we extend the operational semantics of Section 6.2 to account for nondeterministic choice. A command c is now executed with respect to a refiner

$$\text{(SEQ-2)}\quad \frac{(c_0, \sigma, \psi, t, \omega) \longrightarrow (c_0', \sigma', \psi', t', \omega)}{(c_0; c_1, \sigma, \psi, t, \omega) \longrightarrow (c_0'; c_1, \sigma', \psi', t', \omega)}$$

$$\text{(CHOICE)}\quad \frac{head(\psi(\tau)) = i}{(c_0 \ [\!]_\tau \ c_1, \sigma, \psi, t, \omega) \longrightarrow (c_i, \sigma, \psi[\tau := tail(\psi(\tau))], t, \omega)}$$

Figure 6.2: Operational semantics for nondeterministic choice.

ψ, in addition to a state σ, trace t, and joint strategy ω. We thus modify configurations to be 5-tuples $(c, \sigma, \psi, t, \omega)$; terminal configurations now have the form $(\textsf{skip}, \sigma, \psi, t, \omega)$.

All of the operational rules from Figure 6.1 are adapted in the obvious way to handle the new configurations. The only interesting change is SEQ-2, which is restated in Figure 6.2. Nondeterministic choice is evaluated by the new rule CHOICE, which uses refiner ψ to resolve the choice and specifies how the refiner changes as a result. Refiner $\psi[\tau := tail(\psi(\tau))]$ is the refiner ψ with the refinement list for τ replaced by $tail(\psi(\tau))$.

Note that a refiner factors out all nondeterminism in the program: once a refiner, state, and joint strategy have been fixed, execution is completely determined.

6.3.3 A security condition for nondeterministic programs

A well-known problem arises with nondeterministic programs: they are vulnerable to *refinement attacks*, in which a seemingly secure program can be refined to an insecure program. For example, whether the input from H is kept secret in the following program depends on how the nondeterministic choice is resolved:

$$P_3: \quad \textsf{input}\ x\ \textsf{from}\ H; \\ \textsf{output}\ 0\ \textsf{to}\ L\ [\!]\ \textsf{output}\ 1\ \textsf{to}\ L$$

If the choice is made independently of the current state of the program, say by tossing a coin, the program is secure. But if the choice is made as a function of x, the program may leak information about the high input.

To ensure that a program is resistant to refinement attacks, we insist that, for all possible resolutions of nondeterminism, the program does not leak any confidential information. Our model allows this quantification to be expressed cleanly, since refiners encapsulate the resolution of nondeterministic choice. We adapt the security condition of Section 6.2.2 to ensure that, for any refinement of the program, users with access only to channel L do not learn anything about the strategies employed by users of channel H.

Definition 29 (Noninterference Under Refinement) *A command c satisfies* noninterference under refinement *if for all configurations $m = (c, \sigma, \psi, \langle\rangle, \omega)$ and $m' = (c, \sigma, \psi, \langle\rangle, \omega')$ such that $\omega(L) = \omega'(L)$ and for all traces t such that $m \rightsquigarrow t$, there exists a t' such that $t \restriction L = t' \restriction L$ and $m' \rightsquigarrow t'$.*

Some implications of this definition are discussed below.

Low-observable nondeterminism. This security condition rules out refinement attacks but allows programs that appear nondeterministic to a low user. For example, Program P_3 (with $[]$ replaced by $[]_L$) satisfies noninterference under refinement, yet repeated executions may reveal different program behavior to the low user.

Initial refinement lists. The security condition does not require the secrecy of the initial refinement list for H. More concretely, the program

$$\textsf{output } 0 \textsf{ to } L \;\; []_H \;\; \textsf{output } 1 \textsf{ to } L$$

is considered secure even though it reveals information about the first value of $\psi(H)$. The definition thus reflects our intuition that high users interact with the

system only via input and output events on the high channel, which gives them no control over refinement lists. The definition of noninterference under refinement could be adapted to systems where high users may exert control over refinement lists.

Expressivity of refiners. Our model can represent only those refinements that appear as if they were made before the program began execution. Refinements that may depend upon dynamic factors, such as the values of variables or the current program counter, cannot be represented. Our model therefore captures *compiler-time nondeterminism* but not *runtime nondeterminism* [45]. We leave development of more sophisticated refiners as future work.

6.4 Probabilistic programs

Probabilistic choice can be seen as refinement of arbitrary nondeterministic choice. Now that we have shown how refiners can be used to factor out the nondeterministic choices to which we are unable or unwilling to assign probabilities, we can model probabilistic choice explicitly.

We begin by extending the nondeterministic language of Section 6.3 with probabilistic choice:

$$c \quad ::= \quad \ldots \quad | \quad c_0 \; {}_p[\!] \; c_1$$

Informally, probabilistic choice $c_0 \; {}_p[\!] \; c_1$ executes command c_0 with probability p and command c_1 with probability $1 - p$. The probability annotation p must be a real number such that $0 \leq p \leq 1$. We assume that probabilistic choices are made independently of one another.

(PROB-1)

$$\frac{}{(c_0 \ {}_p[\!] \quad c_1, \sigma, \psi, t, \omega) \xrightarrow{p} (c_0, \sigma, \psi, t, \omega)}$$

(PROB-2)

$$\frac{}{(c_0 \ {}_p[\!] \quad c_1, \sigma, \psi, t, \omega) \xrightarrow{1-p} (c_1, \sigma, \psi, t, \omega)}$$

Figure 6.3: Operational semantics for probabilistic choice.

6.4.1 Operational semantics

To incorporate probability in the operational semantics we extend the small-step relation $\longrightarrow$ of previous sections to include a label for probability. We denote membership in the new relation by

$$m \xrightarrow{p} m',$$

meaning that configuration m steps with probability p to configuration m'. Configurations remain unchanged from the nondeterministic language of Section 6.3. The new operational rules defining this relation are given in Figure 6.3. To facilitate backwards-compatibility with the operational rules of previous sections, we interpret $m \longrightarrow m'$ as shorthand for $m \xrightarrow{1} m'$. The operational rules previously given in Figures 6.1 and 6.2 thus remain unchanged.

6.4.2 A probabilistic security condition

It is well-known that probabilistic programs may be secure with respect to nonprobabilistic definitions of noninterference but leak confidential information with high

probability. As an example, consider the following program:

$$
\begin{array}{ll}
P_4: & \textbf{input } x \textbf{ from } H; \\
& \textbf{if } x \textbf{ mod } 2 = 0 \textbf{ then} \\
& \quad \textbf{output } 0 \textbf{ to } L \;{}_{0.99}[]\; \textbf{output } 1 \textbf{ to } L \\
& \textbf{else} \\
& \quad \textbf{output } 0 \textbf{ to } L \;{}_{0.01}[]\; \textbf{output } 1 \textbf{ to } L
\end{array}
$$

If we regard probabilistic choice ${}_p[]$ as identical to nondeterministic choice $[]_L$, then this program satisfies noninterference under refinement. Yet with high probability, the program leaks the parity of the high input to channel L.

Toward preventing such *probabilistic information flows*, observe that if a low trace t is likely to be emitted with one high user strategy and unlikely with another, then the low user learns something about the high strategy by observing the occurrence of t. We thus conclude that our security condition should require that the probability with which low traces are emitted be independent of the strategy employed on the high channel, that is, that low-equivalent configurations should produce particular low traces with the same probability. This intuition is consistent with security conditions given by Gray and Syverson [32] and with the definition of "no evidence" given in Chapter 3.

More formally, let $\mathcal{E}_m(t)$ represent the event that configuration m emits low trace t. Suppose that we had a probability μ_m on such events. Then our security condition should require, for all configurations m and m' that are equivalent except for the choice of high strategy, and all low traces t, that $\mu_m(\mathcal{E}_m(t)) = \mu_{m'}(\mathcal{E}_{m'}(t))$. The remainder of this section is devoted to defining μ_m and $\mathcal{E}_m(t)$.

We begin with two additional intuitions. First, since probabilistic choices are made independently, the probability of an *execution sequence*

$$m_0 \xrightarrow{p_0} m_1 \xrightarrow{p_1} \ldots \xrightarrow{p_{n-1}} m_n$$

is equal to the product of the probabilities p_i of the individual steps. Second, a configuration m could emit the same trace t along multiple sequences, so the probability that m emits t should be the sum of the probabilities associated with those sequences.

Based on these intuitions, we now construct probability measure μ_m by adapting a standard approach for reasoning about probabilities on trees [36]. For any configuration m, relation $\xrightarrow{p}$ gives rise to a rooted directed *probability tree* whose vertices are labeled with configurations, edges are labeled with probabilities, and root is m. Denote the probability tree for m by $\mathcal{T}_m$ and the set of vertices of $\mathcal{T}_m$ by $\mathcal{V}_m$. A *path* in the tree is a sequence of vertices, starting with the root, where each successive pair of vertices is an edge. Given a vertex v, let $tr(v)$ be the trace of events in the configuration with which v is labeled. We say that t *appears at* v when $tr(v) = t$ but $tr(v') \neq t$ for all ancestors v' of v. Let $ap(t)$ be the set of vertices where t appears. In accordance with the intuitions described above, let $\pi(v)$ be the product of the probabilities on the path to v.

A *ray* is an infinite path or a finite path whose terminal node has no descendants, Rays therefore represent maximal execution sequences. Let $\mathcal{R}_m$ denote the set of rays of $\mathcal{T}_m$. Let $R_m(v)$ be the set of rays that go through vertex v:

$$R_m(v) \quad \triangleq \quad \{\rho \in \mathcal{R}_m \,|\, v \text{ is on } \rho\}.$$

Let $\mathcal{A}_m$ be the σ-algebra on $\mathcal{R}_m$ generated by sets of rays going through particular vertices, that is, by the set $\{R_m(v) \,|\, v \in \mathcal{V}_m\}$.The following result yields a probability measure on sets of rays. It is a consequence of elementary results in probability theory, and we omit the proof.

Theorem 9 *For any configuration m, there exists a unique probability measure*

μ_m on $\mathcal{A}_m$ such that for all $v \in \mathcal{V}_m$ we have $\mu_m(R_m(v)) = \pi(v)$.

Now that we have constructed μ_m, we must show how to use it to obtain the probability of a set of traces in terms of the probability of a corresponding set of rays. For a set T of traces, let $R_m(T)$ be the set of rays on which a trace in T appears. Let $em_m(T) = \{t \in T \mid m \rightsquigarrow t\}$ be the set of traces in T emitted by m, and note that

$$R_m(T) \quad \triangleq \quad \bigcup_{t \in em_m(T)} \bigcup_{v \in ap(t)} R_m(v),$$

because a trace appears on a ray r if and only if it appears at a vertex v on r. The set $R_m(T)$ is measurable with respect to $\mathcal{A}_m$ because both $em_m(T)$ and $\mathcal{V}_m$ are countable sets. Given a trace t, the set $\{R_m(v) \mid v \in ap(t)\}$ is a partition of the set of rays on which t appears. It follows that

$$\begin{aligned} \mu_m(R_m(\{t\})) &= \mu_m(\textstyle\bigcup_{v \in ap(t)} R_m(v)) \\ &= \textstyle\sum_{v \in ap(t)} \mu_m(R_m(v)) \\ &= \textstyle\sum_{v \in ap(t)} \pi(v), \end{aligned}$$

that is, that the probability that m emits t is equal to the sum of the values $\pi(v)$ for vertices v where t appears, as desired.

We can now define $\mathcal{E}_m(t)$. Given a security type τ and a trace t, let $[t]_\tau$ be the equivalence class of traces that are equal to t when restricted to τ:

$$[t]_\tau \quad \triangleq \quad \{t' \in \mathbf{Tr} \mid t' \restriction \tau \, = \, t \restriction \tau\}.$$

Finally, let $\mathcal{E}_m(t)$ be the set of rays on which there is some vertex v such that $tr(v) \restriction L = t \restriction L$:

$$\mathcal{E}_m(t) \quad \triangleq \quad R_m([t]_L).$$

The set $\mathcal{E}_m(t)$ is in $\mathcal{A}_m$. By Theorem 9, $\mu_m(\mathcal{E}_m(t))$ is equal to the sum of values

$\pi(v)$ for vertices v such that $tr(v) \restriction L = t \restriction L$ and $tr(v') \restriction L \neq t \restriction L$ for any ancestor v' of v.

We are now ready to formalize our security condition.

Definition 30 (Probabilistic Noninterference) *A command c satisfies* probabilistic noninterference *if for all $m = (c, \sigma, \psi, \langle\rangle, \omega)$ and $m' = (c, \sigma, \psi, \langle\rangle, \omega')$ such that $\omega(L) = \omega'(L)$ and for all $t \in \mathbf{Tr}(\mathbf{Ev}(L))$, we have $\mu_m(\mathcal{E}_m(t)) = \mu_{m'}(\mathcal{E}_{m'}(t))$.*

Returning to Program P_4 at the start of this section, it is easy to check that the probability of the low trace $\langle out(L, 0)\rangle$ is 0.99 when the high strategy is to input an even number, and 0.01 when the high strategy is to input an odd number. Clearly, the program does not satisfy probabilistic noninterference.

Program P_1, the insecure one-time pad implementation, does not satisfy probabilistic noninterference. However, if the output to H is removed, the resulting program

while (**true**) **do**
  $x := 0 \ {}_{0.5}[] \ x := 1$;
  input y **from** H;
  output x **xor** $(y$ **mod** $2)$ **to** L

does satisfy noninterference. The probability of low outputs is independent of the high strategy, which can no longer exploit knowledge of the value of the one-time pad x.

User strategies as defined thus far are deterministic. However, our approach to reasoning about probability applies to randomized user strategies as well as to randomized programs, so it would be straightforward to adapt our model to handle randomized strategies.

6.5 Characterizing noninterference as secrecy

It is straightforward to interpret our definitions of noninterference as instances of the more general definitions of secrecy presented in Chapter 3. As in Chapter 5, we must demonstrate how a system—in this case, a program—gives rise to a set of runs. We reason about two agents L and H who interact with the channels of the same names, and for simplicity we assume that a state σ and a refiner ψ are fixed. (If they were not fixed, we could, without loss of generality, model the states of agents to include information about the initial state and the refiner ψ. In particular, by including the initial state and the refiner in L's local state, we can ensure that H maintains secrecy with respect to L even if L knows both the initial values of variables and how the nondeterministic choices in the program will be resolved.) We also assume that an agent's local state encodes the strategy that she is using. This is technically necessary, because the goal is to ensure that low agents obtain no information about which strategy a high agent is using, but it also seems reasonable to assume that agents know how they will behave.

Our goal is to exhibit a set of runs that characterizes the possible execution sequences of the system. Given a particular joint strategy ω, we can derive a natural set of runs from the corresponding rays of the probability tree for the configuration $(c, \sigma, \psi, t, \omega)$; a system can then be characterized by the sets of runs that arise from the possible joint strategies.

We associate a run r^ρ with a ray ρ as follows. If ρ is infinite, let r^ρ be a run that, for each time step n, maps to a global state $\langle (t \restriction L, \omega_L), (t \restriction H, \omega_H) \rangle$, where t is the trace appearing at the depth-n vertex of ρ and ω_L, ω_H are user strategies.[1]

[1]We apologize for the inevitable overloading of the symbol m, which denotes time steps in previous chapters and configurations in this chapter. In this section we use n for time steps and m for configurations.

(The first element of the pair is the local state of the low agent, and the second element is the local state of the high agent.) More formally, let $t(\rho, n)$ be the trace t such that the depth-n vertex of ρ is labeled with configuration $(c, \sigma, \psi, t, \omega)$. Given an infinite ray ρ from the probability tree $\mathcal{T}_{(c,\sigma,\psi,t,\omega)}$, let r^ρ be the run such that $r^\rho(n) = \langle (t(\rho, n) \restriction L, \omega(L)), (t(\rho, n) \restriction H, \omega(H)) \rangle$. Given a finite ray ρ whose terminal vertex is of depth N, let r^ρ be the run such that

$$r^\rho(n) = \langle (t(\rho, n) \restriction L, \omega(L)), (t(\rho, n) \restriction H, \omega(H)) \rangle$$

for all $n \leq N$, and

$$r^\rho(n) = \langle (t(\rho, N) \restriction L, \omega(L)), (t(\rho, N) \restriction H, \omega(H)) \rangle$$

for all $n \geq N$.

A joint strategy ω thus gives us a set of runs D_ω, a corresponding σ-algebra $\mathcal{F}_\omega$, and a probability measure μ_ω. (Since there is a one-to-one correspondence between rays and runs, $\mathcal{F}_\omega$ contains exactly those sets of runs derived from sets of rays in $\mathcal{A}_{(c,\sigma,\psi,t,\omega)}$.) Let $\mathcal{D} \triangleq \{D_\omega \,|\, \omega \in \mathbf{Strat}\}$, and let $\Delta \triangleq \{(D_\omega, \mathcal{F}_\omega, \mu_\omega,) \,|\, D_\omega \in \mathcal{D}\}$, in accordance with the definition of an adversarial probability system given in Section 3.2.4. Finally, let $\mathcal{R} \triangleq \cup_{D \in \mathcal{D}} D$.

This setup is sufficient to establish a connection between probabilistic noninterference and "no evidence" (in the sense of Definition 13):

Theorem 10 *Command c satisfies probabilistic noninterference if and only if, for all initial states σ and refiners ψ, the low agent L obtains no evidence for the initial choice in the adversarial probability system $(\mathcal{R}, \mathcal{D}, \Delta)$ derived from c, σ, and ψ.*

Proof: Given a point (r, n), let $t_L(r, n)$ be the event trace appearing in the local state $r_L(n)$ of the low user. We begin by noting that for every joint strategy ω

and every point (r, n) such that $\mathcal{R}(\mathcal{K}_L(r, n)) \cap D_\omega \neq \emptyset$, we have $\mu_\omega(\mathcal{R}(\mathcal{K}_L(r, n))) = \mu_m(\mathcal{E}_m(t_L(n)))$ (where $m = (c, \sigma, \psi, t, \omega)$). In other words, the probability of a low-information set is the same as the probability that the associated low trace will be emitted by the configuration using joint strategy ω. This follows because we have

$$\mathcal{R}(\mathcal{K}_L(r, n)) = \{r^\rho \mid \rho \in \mathcal{E}_m(t_L(n)),$$

by the definitions of $\mathcal{R}(\mathcal{K}_L(r, n))$ and $\mathcal{E}_m(t_L(n))$. Now, note that given a point (r, n) and partitions $D_\omega, D_{\omega'} \in \mathcal{D}$, we have

$$\mathcal{R}(\mathcal{K}_L(r, n)) \cap D_\omega \neq \emptyset$$

and

$$\mathcal{R}(\mathcal{K}_L(r, n)) \cap D_{\omega'} \neq \emptyset$$

only if $\omega(L) = \omega'(L)$, because the local state of the low agent includes the low strategy. The reader can now check that Definitions 13 and 30 coincide. □

This result may seem tedious and relatively trivial, but by translating programs to sets of runs we can exploit Theorem 5 to demonstrate that probabilistic noninterference coincides with run-based secrecy (according to Definition 12). In particular, for any countable set of joint strategies, probabilistic noninterference implies that the high agent maintains probabilistic run-based f_{prot}-secrecy with respect to the low-agent (where f_{prot} extracts the high agent's strategy) in the corresponding generalized run-based probability system. This result can be restated in terms of the low agent's *beliefs* about which strategy the high agent is using: if the low user starts out with an initial probability distribution on high strategies, he will be unable to improve the accuracy of that distribution as he interacts with the system.

6.6 A sound type system

The problem of characterizing programs that satisfy noninterference is, for many definitions of noninterference, intractable. For definitions appearing in the previous sections, there is a straightforward reduction from the halting problem to the noninterference problem. It follows that no decision procedure for certifying the information-flow security of programs can be both sound and complete with respect to our definitions of noninterference. The goal of this section is to demonstrate that static analysis techniques can be used to soundly identify secure programs.

We use a type system based on that of Volpano, Smith, and Irvine [92]. It is interesting to note that a type system designed to enforce batch-job noninterference conditions also enforces our interactive conditions, including probabilistic noninterference, even though the type system is oblivious to the subtleties of probability, interactivity, and user strategies. We believe that other type systems for information flow (e.g., [4, 48, 84, 86]) can also be easily adapted for our interactive model, and thus that advances in precision and expressiveness can be applied to our work.

The type system consists of a set of axioms and inference rules for deriving *typing judgments* of the form $\Gamma \vdash p : \kappa$, meaning that phrase p has phrase type κ under variable typing Γ. A *phrase* is either an expression or a command. A *phrase type* is either a security type τ or a command type τ *cmd*, where $\tau \in \mathcal{L}$. A *variable typing* is a function $\Gamma : \mathbf{Var} \to \mathcal{L}$ mapping from variables to security types. Informally, a command c has type τ *cmd* when τ is a lower-bound on the effects that c may have, that is, when the types (under Γ) of any variables that c updates are bounded below by τ, and any input or output that c performs is on channels whose security type is bounded below by τ.

$$\text{(T-Lit)}\ \frac{}{\Gamma \vdash n : \tau} \qquad \text{(T-Var)}\ \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \text{(T-Op)}\ \frac{\Gamma \vdash e_0 : \tau \quad \Gamma \vdash e_1 : \tau}{\Gamma \vdash e_0 \oplus e_1 : \tau}$$

$$\text{(T-Assign)}\ \frac{\Gamma(x) = \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash x := e : \tau\ cmd} \qquad \text{(T-Skip)}\ \frac{}{\Gamma \vdash \textsf{skip} : \tau\ cmd}$$

$$\text{(T-If)}\ \frac{\Gamma \vdash e : \tau \quad \Gamma \vdash c_0 : \tau\ cmd \quad \Gamma \vdash c_1 : \tau\ cmd}{\Gamma \vdash \textsf{if}\ e\ \textsf{then}\ c_0\ \textsf{else}\ c_1 : \tau\ cmd}$$

$$\text{(T-Seq)}\ \frac{\Gamma \vdash c_0 : \tau\ cmd \quad \Gamma \vdash c_1 : \tau\ cmd}{\Gamma \vdash c_0; c_1 : \tau\ cmd}$$

$$\text{(T-While)}\ \frac{\Gamma \vdash e : L \quad \Gamma \vdash c : \tau\ cmd}{\Gamma \vdash \textsf{while}\ e\ \textsf{do}\ c : L\ cmd} \qquad \text{(T-Choice)}\ \frac{\Gamma \vdash c_0 : \tau\ cmd \quad \Gamma \vdash c_1 : \tau\ cmd}{\Gamma \vdash c_0\ [\!]_\tau\ c_1 : \tau\ cmd}$$

$$\text{(T-Prob)}\ \frac{\Gamma \vdash c_0 : \tau\ cmd \quad \Gamma \vdash c_1 : \tau\ cmd}{\Gamma \vdash c_0\ {}_p[\!]\ \ c_1 : \tau\ cmd}$$

$$\text{(T-In)}\ \frac{\Gamma(x) = \tau' \quad \tau \leq \tau'}{\Gamma \vdash \textsf{input}\ x\ \textsf{from}\ \tau : \tau\ cmd} \qquad \text{(T-Out)}\ \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \textsf{output}\ e\ \textsf{to}\ \tau : \tau\ cmd}$$

$$\text{(T-Subtype)}\ \frac{\Gamma \vdash p : \kappa_0 \quad \kappa_0 \leq \kappa_1}{\Gamma \vdash p : \kappa_1}$$

$$\text{(ST-Base)}\ \frac{}{L \leq H} \qquad \text{(ST-Refl)}\ \frac{}{\kappa \leq \kappa} \qquad \text{(ST-Cmd)}\ \frac{\tau_0 \leq \tau_1}{\tau_1\ cmd \leq \tau_0\ cmd}$$

Figure 6.4: Typing rules.

Axioms and inference rules for the type system are given in Figure 6.4. There are two types of rules: typing rules (prefixed with "T") and subtyping rules (prefixed with "ST"). Typing rules can be used to infer the type of an expression or command directly. Subtyping rules allow a low-typed expression to be treated as a high-typed expression and a high-typed command to be treated as a low-typed command. (It is safe, for example, to store a low-typed expression in a high variable, or to output data to a high user in the body of a loop with a low-typed guard.)

Most of the rules in this type system are standard. Rules T-In and T-Out are both similar to T-Assign: T-In ensures that values read from the τ channel are stored in variables whose type is bounded below by τ, whereas T-Out ensures that only τ-typed expressions are output on the τ channel. Rules T-Choice and T-Prob are similar to T-Seq, except that T-Choice also checks that the typing is consistent with the syntactic type annotation. Rule T-While forbids high-guarded loops, ensuring that loop termination does not depend on the high user's strategy. This prohibits insecure programs such as P_2 (in Section 6.2.2). We believe this rule could be relaxed using techniques described by Boudol and Castellani [4] and Smith [86].

The following theorem states that this type system soundly enforces noninterference. Recall that our security conditions do not depend on the security types of variables. Noninterference is enforced provided there exists some variable typing under which the program is well-typed.[2] The proof is in Appendix C.

Theorem 11 For any command c, if there exists a variable typing Γ and a security

[2]Because the security types of variables can be inferred, programmers need not specify them. In a (trivially secure) program with no high inputs, for example, all variables can be assigned type L.

type τ such that $\Gamma \vdash c : \tau\ cmd$, then

(a) if c does not contain nondeterministic or probabilistic choice, then c satisfies noninterference;

(b) if c does not contain probabilistic choice, then c satisfies noninterference under refinement; and

(c) c satisfies probabilistic noninterference.

6.7 Related work

Definitions of information-flow security for imperative programs began with the work of Denning and Denning [13]. Many subsequent papers define information-flow security for various sequential imperative languages, but nearly all of these papers assume a batch-job model of computation. Therefore, they attempt to ensure the secrecy of high-typed program variables rather than of the behavior of high users who interact with the system. See Sabelfeld and Myers [76] for a survey of language-based information-flow security.

Another line of work considers end-to-end information-flow restrictions for nondeterministic systems that provide input and output functionality for users. Definitions of noninterference exist both for abstract systems (such as finite state machines) that include input and output operations (Goguen and Meseguer [31], McCullough [56], McLean [58], Mantel [54]), and for systems described using process algebras such as CCS, the π-calculus, and related formalisms (Focardi and Gorrieri [22], Ryan and Schneider [73], Zdancewic and Myers [96]).

Definitions of noninterference based on process algebras typically require that the observations made by a public user are the same regardless of which high processes (if any) are interacting with the system. These definitions are thus

similar in spirit to our definitions of noninterference. Indeed, there is a close connection between strategies and processes: both can be viewed as description of how an agent will behave in an interactive setting. A formal comparison with process-based definitions (such as [24]) may uncover further connections between process-based system models and imperative programs.

Wittbold and Johnson [93] give the first strategy-based definition of information-flow security, and Gray and Syverson [32] give a strategy-based definition of probabilistic noninterference. Our definitions of noninterference are the first strategy-based security conditions for an imperative programming language of which we are aware. Our work can thus be viewed as a unification of two distinct strands of the information-flow literature. In this sense our work is similar to that of Mantel and Sabelfeld [55], who demonstrate a connection between security predicates taken from the *MAKS* framework of Mantel [54] and bisimulation-based definitions of security for a concurrent imperative language due to Sabelfeld and Sands [77]. However, Mantel and Sabelfeld do not consider interactive programs.

Our probabilistic noninterference condition can be interpreted as precluding programs that allow low users to make observations that improve the accuracy of their beliefs about high behavior, that is, their beliefs about which high strategy is used. As we discuss in Section 6.5, probabilistic noninterference suffices to ensure that low users cannot improve the accuracy of their subjective beliefs about high behavior by interacting with a program. Our probabilistic security condition also ensures that the quantity of information flow due to a secure program is exactly zero bits in the belief-based quantitative information-flow model of Clarkson, Myers, and Schneider [10].

The bisimulation-based security condition of Sabelfeld and Sands [77] can be

viewed as a relaxation of the batch-job model. However, as Mantel and Sabelfeld [55] point out, bisimulation-based definitions are difficult to relate to trace-based conditions when a nondeterministic choice operator is present in the language. The following program, for example, satisfies both noninterference under refinement and probabilistic noninterference (for suitable interpretations of the $[]$ operator), but it is not secure with respect to a bisimulation-based definition of security:

```
input x from H;
if (x = 0)
   output 0 to L;
   {output 1 to L  []  output 2 to L}
else
   {output 0 to L; output 1 to L} []
   {output 0 to L; output 2 to L}
```

Bisimulation-based security conditions implicitly assume that users can observe internal choices made by a program. When users observe only inputs and outputs on channels, our observational model is more appropriate.

Interactivity between users and a program is similar to message-passing between threads. Sabelfeld and Mantel [75] present a multi-threaded imperative language with explicit send, blocking receive, and non-blocking receive operators for communication between processes. They describe a bisimulation-based security condition and a type system to enforce it. However, it is not clear how to model user behavior in their setting. Users cannot be modeled as processes since user behavior is unknown, and their security condition applies only if the entire program is known.

Almeida Matos, Boudol, and Castellani [2] state a bisimulation-based security condition for *reactive programs*, which allow limited communication between processes, and they give a sound type system to enforce the condition. In their

language, programs react to the presence and absence of named broadcast signals and can emit signals to other programs in a "local area." It is possible to implement our higher-level channels and events within a local area, using their lower-level reactivity operators. However, it is unclear how to use reactivity to model interactions with unknown users who are not part of a local area.

Focardi and Rossi [23] study the security of processes in *dynamic contexts* where the environment, including high processes, can change throughout execution. This is similar to how high user strategies describe changing inputs throughout execution. However, user strategies depend upon the history of the computation, whereas dynamic contexts do not, so it is unclear how to encode a user strategy using dynamic contexts.

Previous work dealing with the susceptibility of possibilistic noninterference to refinement attacks takes one of two approaches to specifying how nondeterministic choice is resolved. One approach is to assume that choices are made according to fixed probability distributions, as we do in Section 6.4. Volpano and Smith [91], for example, describe a scheduler for a multithreaded language that chooses threads to execute according to a uniform probability distribution. A second approach is to insist that programs be *observationally deterministic* for low users. McLean [57] and Roscoe [72] both advocate observational determinism as an appropriate security condition for nondeterministic systems, and Zdancewic and Myers [96] give a security condition based on observational determinism for a concurrent language based on the join calculus [25].

Observational determinism implies noninterference under refinement and thus immunity to refinement attacks. In settings where the resolution of nondeterministic choice may depend on confidential information, we conjecture that observational

determinism and noninterference under refinement are equivalent. However, when the resolution of some choices is independent of confidential information, observational determinism is a stronger condition: any program that is observationally deterministic satisfies noninterference under refinement, but the converse does not hold.

Chapter 7

Conclusion

We have defined general notions of secrecy for systems with which multiple agents interact over time, and have given syntactic characterizations of our definitions that connect them to logics of knowledge and probability. We have applied our definitions to the problem of characterizing the absence of information flow, and have shown how our definitions can be viewed as a generalization of a variety of information-flow definitions that have been proposed in the past. We have also given general definitions of anonymity for agents acting in multiagent systems, and have compared and contrasted our definitions to other similar definitions of anonymity.

Our knowledge-based system framework provides a number of advantages:

- We are able to state confidentiality properties succinctly and intuitively, and in terms of the knowledge of the observers or attackers who interact with the system.

- Our system has a well-defined semantics that lets us reason about knowledge in systems of interest, such as systems specified using process algebras or strand spaces as well as systems derived from imperative programs.

- We are able to give straightforward probabilistic definitions of secrecy and anonymity.

As we discuss in Chapter 5, we are not the first to attempt to provide a general framework for analyzing secrecy or anonymity. However, we believe that our definitions are more closely related to the intuitions that people in the field have

had, in part because those intuitions have often often been expressed informally in terms of the knowledge of the agents who interact with a system.

Although we have discussed secrecy largely with respect to the kinds of input and output systems that have been popular with the theoretical security community, our definitions of secrecy apply in other contexts, such as protocol analysis, semantics for imperative programming languages, and database theory. Chor, Goldreich, Kushilevitz, and Sudan [8], for example, consider the situation where a user wants to query a replicated database for some specific database item, but wants a guarantee that no one will be able to determine, based on his query, which item he wants. It is not hard to show that the definition of privacy given by Chor et al. is a special case of secrecy in an adversarial system with a cell corresponding to each possible item choice.

One possible direction for future work is a careful consideration of how definitions of secrecy can be weakened to make them more useful in practice. Here we briefly consider some of the issues involved:

- *Declassification:* Not all facts can be kept secret in a real-world computer system. The canonical example is password checking, where a system is forced to release information when it tells an attacker that a password is invalid. Declassification for information-flow properties has been addressed by, for example, Myers, Sabelfeld, and Zdancewic [65]. It would be interesting to compare their approach to our syntactic approach to secrecy, keeping in mind that our syntactic definitions can be easily weakened simply by removing facts from the set of facts that an agent is required to think are possible.

- *Computational secrecy:* Our definitions of secrecy are most appropriate for

attackers with unlimited computational power, since agents "know" any fact that follows logically from their local state, given the constraints of the system. Such an assumption is unreasonable for most cryptographic systems, where secrecy depends on the inability of attackers to solve difficult computational problems. The process-algebraic approach advocated by Mitchell, Ramanathan, Scedrov, and Teague [63] and the work on probabilistic algorithm knowledge of Halpern and Pucella [43] may help to shed light on how definitions of secrecy can be weakened to account for agents with computational limitations.

- *Quantitative secrecy:* Our definitions of probabilistic secrecy require absolute secrecy: no information about the state of high users may be revealed to low users. One way to weaken these requirements is to place bounds on the amount of information (measured in, say, bits) that can be revealed to low users. This intuition can be formalized using the information-theoretic notions of relative entropy and mutual information, which can be viewed as generalizations of probabilistic independence. Rather than requiring that a low agent learns nothing, quantitative definitions of secrecy use entropy-based metrics to measure (and place bounds on) how much a low agent learns as he interacts with a system. Information-theoretic approaches to secrecy have been discussed by Wittbold and Johnson [93], and more recently by Clark, Hunt, and Malacaria [9], Lowe [52], Di Pierro, Hankin, and Wiklikcy [14], and Clarkson, Myers, and Schneider [10].

- *Statistical privacy:* In some systems, such as databases that release aggregate statistical information about individuals, our definitions of secrecy are

much too strong because they rule out the release of any useful information. Formal definitions of secrecy and privacy for such systems have recently been proposed by Evfimievski, Gehrke, and Srikant [18] and by Chawla, Dwork, McSherry, Smith and Wee [6]. These definitions seek to limit the information that an attacker can learn about a user whose personal information is stored in the database. It would be interesting to cast those definitions as weakenings of secrecy.

These weakenings of secrecy are all conceptually different, but there are obviously many relations and connections among them. We hope that our work will help to clarify some of the issues involved.

With respect to anonymity, there are a number of issues that this paper has not addressed. We have focused almost exclusively on properties of anonymity and have not considered related notions, such as *pseudonymity* and *unlinkability* [47, 68]. There seems to be no intrinsic difficulty capturing these notions in our framework. For example, one form of message unlinkability specifies that no two messages sent by an anonymous sender can be "linked," in the sense that an observer can determine that both messages were sent by the same sender. More formally, two actions a and a' are linked with respect to an observer o if o knows that there exists an agent i who performed both a and a'. This definition can be directly captured using knowledge. Its negation says that o considers it possible that there exist two distinct agents who performed a and a'; this can be viewed as a definition of *minimal unlinkability*. This minimal requirement can be strengthened, exactly as our definitions of anonymity were, to include larger numbers of distinct agents, probability, and so on. Although we have not worked out the details, we believe that our approach will be similarly applicable to other related definitions.

Another obviously important issue is verification and enforcement, that is, checking whether a given system specifies one of the confidentiality properties we have introduced. The sound type system of Section 6.6 is one example of a static enforcement mechanism, but is specific to the domain of imperative programs. Another technique for enforcement that is more generally applicable to our definitions is model-checking. Recent work on the problem of model checking in the multiagent systems framework suggests that this may be viable. Van der Meyden [59] discusses algorithms and complexity results for model checking a wide range of epistemic formulas in the runs and systems framework, and van der Meyden and Su [60] use these results to verify the dining cryptographers protocol [5], using formulas much like those described in Section 4.1.3. Even though model checking of formulas involving knowledge seems to be intractable for large problems, these results are a promising first step toward being able to use knowledge for both the specification and verification of secrecy and anonymity. Shmatikov [83], for example, analyzes the Crowds system using the probabilistic model checker PRISM [49]. This is a particularly good example of how definitions of anonymity can be made precise using logic and probability, and how model-checking can generate new insights into the functioning of a deployed protocol.

Our definitions of noninterference for imperative programs constitute a step toward understanding and enforcing information-flow security in real-world programs. Many programs interact with users, and the behavior of these users will often be dependent on previous inputs and outputs. Also, many programs, especially servers, are intended to run indefinitely rather than to perform some computation and then halt. Our model of interactivity is thus more suitable for analyzing real-world systems than the batch-job model. In addition, our imperative language

approximates the implementation of real-world interactive programs more closely than abstract system models such as the π-calculus. Our work thereby contributes to understanding the security properties of programs written in languages with information flow control, such as Jif [66] or Flow Caml [85], that support user input and output.

In closing, we want to emphasize that our goal is not to advocate any particular definitions of secrecy or anonymity as being the "right" definition for all situations. Rather, we argue that system designers should work with definitions that are as simple and intuitive as possible. The definitions and results that we have presented, and their underlying intuitions of knowledge and independence, do not depend on the particular system representation that we describe here, so they should be broadly applicable. Indeed, one major theme of our work is the importance of having good system models and of isolating general notions of confidentiality from particular system representations. Given the right system model and the right measure of uncertainty, a reasonable definition of secrecy or anonymity usually follows quite easily. By providing a general, straightforward way to model systems, the runs-and-systems framework provides a useful foundation for appropriate definitions of security.

Appendix A

Proofs for Chapter 3

A.1 Examples of systems

In this section, we give examples of simple systems that show the limitations of various theorems. All the systems involve only two agents, and we ignore the environment state. We describe each run using the notation

$$\langle (X_{i,1}, X_{j,1}), (X_{i,2}, X_{j,2}), (X_{i,3}, X_{j,3}), \ldots \rangle,$$

where $X_{i,k}$ is the local state of agent i at time k. For asynchronous systems, we assume that the final global state—$(X_{i,3}, X_{j,3})$, in the example above—is repeated infinitely. For synchronous systems we need different states at each time step, so we assume that global states not explicitly listed encode the time in some way, so change at each time step. For notational simplicity, we use the same symbol for a local state and its corresponding information set.

Example 1: Suppose that the synchronous system $\mathcal{R}$ consists of the following two runs:

$$\begin{aligned} r_1 &\triangleq \langle (X, A), (Y_1, B_1), (Y_2, B_2), \ldots \rangle \\ r_2 &\triangleq \langle (Z, A), (Y_1, C_1), (Y_2, C_2), \ldots \rangle \end{aligned}$$

Note that agent 2 has perfect recall in $\mathcal{R}$, but agent 1 does not (since at time 0 agent 1 knows the run, but at all later times, he does not). It is easy to check that agent 2 maintains synchronous secrecy with respect to 1, but not run-based secrecy, since $\mathcal{R}(B_1) \cap \mathcal{R}(Z) = \emptyset$.

For the same reasons, if we take the probability measure μ on $\mathcal{R}$ with $\mu(r_1) = \mu(r_2) = 1/2$, probabilistic synchronous secrecy and run-based probabilistic secrecy do not coincide. This shows that the perfect recall requirement is necessary in both Propositions 3 and 8. □

Example 2: Suppose that the $\mathcal{R}$ consists of the following three runs (where, in each case, the last state repeats infinitely often):

$$\begin{aligned} r_1 &\triangleq \langle (X, A) \ldots \rangle \\ r_2 &\triangleq \langle (X, B), (Y, A) \ldots \rangle \\ r_3 &\triangleq \langle (Y, A) \ldots \rangle, \end{aligned}$$

It is easy to see that agent 2 maintains run-based secrecy with respect to agent 1 in $\mathcal{R}$, but not total secrecy or synchronous secrecy (since, for example, $Y \cap B = \emptyset$).

Now consider a probability measure μ on $\mathcal{R}$ such $\mu(r_1) = \mu(r_3) = 2/5$, and $\mu(r_2) = 1/5$. Then $\mu(\mathcal{R}(A) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(A) \,|\, \mathcal{R}(Y)) = 1$ and $\mu(\mathcal{R}(B) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(B) \,|\, \mathcal{R}(Y)) = 1/3$, so agent 2 maintains run-based probabilistic secrecy with respect to 1 in $\mathcal{R}$. 1 does not maintain probabilistic secrecy with respect to 2 in $(\mathcal{R}, \mu)$, since $\mu(\mathcal{R}(X) \,|\, \mathcal{R}(A)) = 3/5$, while $\mu(\mathcal{R}(X) \,|\, \mathcal{R}(B)) = 1$. Thus, if the agents do not have perfect recall and the system is not synchronous, then run-based probabilistic secrecy is not necessarily symmetric. □

Example 3: Suppose that the synchronous system $\mathcal{R}$ consists of the following four runs:

$$\begin{aligned} r_1 &\triangleq \langle (X, A), (Y_1, C_1), (Y_2, C_2), \ldots \rangle \\ r_2 &\triangleq \langle (X, B), (Y_1, D_1), (Y_2, D_2), \ldots \rangle \\ r_3 &\triangleq \langle (Q, A), (R_1, D_1), (R_2, D_2), \ldots \rangle \\ r_4 &\triangleq \langle (Q, B), (R_1, C_2), (R_2, C_2), \ldots \rangle \end{aligned}$$

Note that agent 2 does not have perfect recall in $\mathcal{R}$, although agent 1 does. Let μ give each of these runs equal probability. It is easy to check that for all $i \geq 1$, we have

- $\mu(\mathcal{R}(A) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(A) \,|\, \mathcal{R}(Q)) = 1/2$,
- $\mu(\mathcal{R}(B) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(B) \,|\, \mathcal{R}(Q)) = 1/2$,
- $\mu(\mathcal{R}(C_i) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(C_i) \,|\, \mathcal{R}(Q)) = 1/2$, and
- $\mu(\mathcal{R}(D_i) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(D_i) \,|\, \mathcal{R}(Q)) = 1/2$.

Because $\mathcal{R}(X) = \mathcal{R}(Y_i)$ and $\mathcal{R}(Q) = \mathcal{R}(R_i)$ for all $i \geq 1$, it follows that agent 2 maintains run-based probabilistic secrecy with respect to 1 in $(\mathcal{R}, \mu)$.

Now, let p be a primitive proposition and let π be an interpretation such that p is true if 2's local state is either A or D_1. Thus, p is 2-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$. Since $\mu(\mathcal{R}(A) \cup \mathcal{R}(D_1) \,|\, \mathcal{R}(X)) = 1$ while $\mu(\mathcal{R}(A) \cup \mathcal{R}(D_1) \,|\, \mathcal{R}(Q)) = 1/2$, there is no constant σ such that $\mathcal{I} \models \Pr_1(\Diamond p) = \sigma$. This shows that the assumption that agent j has perfect recall is necessary in Theorem 4. □

A.2 Proofs for Section 3.1

Proposition 2: *If $\mathcal{R}$ is a system where i and j have perfect recall, C depends only on timing, and j maintains C-secrecy with respect to i, then j maintains run-based secrecy with respect to i.*

Proof: Given (r, m) and (r', m'), we must find a run r'' and times m_1 and m_2 such that $r''_i(m_1) = r_i(m)$ and $r''_j(m_2) = r'_j(m')$. Because C depends only on timing, there exists a point (r, n) such that $(r', m') \in C(r, n)$. The proof now splits into two cases:

- Suppose that $n \geq m$. By C-secrecy, there exists a point (r'', m_2) such that $r''_i(m_2) = r_i(n)$ and $r''_j(m_2) = r'_j(m')$. Because i has perfect recall, there exists some $m_1 \leq m_2$ such that $r''_i(m_1) = r_i(m)$.

- Suppose that $m > n$. Because C depends only on timing, there exists $n' \geq m'$ such that $(r', n') \in C(r, m)$. By C-secrecy, there exists a point (r'', m_2) such that $r''_i(m_2) = r_i(m)$ and $r''_j(m_2) = r'_j(n')$. Because j has perfect recall, there exists some $m_1 \leq m_2$ such that $r''_j(m_1) = r'_j(m')$.

□

Proposition 3: *If $\mathcal{R}$ is a synchronous system where both i and j have perfect recall, then agent j maintains synchronous secrecy with respect to i iff j maintains run-based secrecy with respect to i.*

Proof: Suppose that agent j maintains synchronous secrecy with respect to j in $\mathcal{R}$. Because both i and j have perfect recall, j maintains run-based secrecy with respect to i by Proposition 2.

Conversely, suppose that j maintains run-based secrecy with respect to i. Given runs $r, r' \in \mathcal{R}$ and any time m, there exists a run r'', and times n and n', such that $r''_i(n) = r_i(m)$ and $r''_j(n') = r'_j(m)$. By synchrony, $m = n = n'$, and we have $r''_i(m) = r_i(m)$ and $r''_j(m) = r'_j(m)$. Thus j maintains synchronous secrecy with respect to i. □

Proposition 4: *A formula ϕ is j-local in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ iff there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \phi$ whenever $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$.*

Proof: Suppose that ϕ is j-local. Let

$$\Omega = \{\mathcal{K}_j(r, m) \mid (\mathcal{I}, r, m) \models \phi\}.$$

If $(\mathcal{I}, r, m) \models \phi$, then $\mathcal{K}_j(r, m) \in \Omega$ by definition, so $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Likewise, if $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$, then $(r, m) \in \mathcal{K}_j(r', m')$ for some (r', m') such that $(\mathcal{I}, r', m') \models \phi$. By j-locality, $(\mathcal{I}, r, m) \models \phi$.

Conversely suppose that there exists a set of j-information sets Ω such that $(\mathcal{I}, r, m) \models \phi$ whenever $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. We need to show that ϕ is j-local. Suppose that $r_j(m) = r'_j(m')$. If $(\mathcal{I}, r, m) \models \phi$, then $(r, m) \in \mathcal{K}_j(r'', m'')$ for some $\mathcal{K}_j(r'', m'') \in \Omega$, and clearly $(r', m') \in \mathcal{K}_j(r'', m'') \subseteq \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$ too, so $(\mathcal{I}, r', m') \models \phi$ by assumption. □

Theorem 1: *Suppose that C is an i-allowability function. Agent j maintains C-secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π and point (r, m), if ϕ is j-local and $(\mathcal{I}, r', m') \models \phi$ for some $(r', m') \in C(r, m)$, then $(\mathcal{I}, r, m) \models P_i\phi$.*

Proof: Suppose that j maintains C-secrecy with respect to i in $\mathcal{R}$. Let π be an interpretation, let (r, m) be a point, and let ϕ be a formula that is j-local such that $(\mathcal{I}, r', m') \models \phi$ for some $(r', m') \in C(r, m)$. By C-secrecy, there exists a point $(r'', m'') \in \mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m')$. Because ϕ is j-local, $(\mathcal{I}, r'', m'') \models \phi$. Thus $(\mathcal{I}, r, m) \models P_i\phi$, as required.

For the converse, given $(r, m) \in \mathcal{PT}(\mathcal{R})$ and $(r', m') \in C(r, m)$, let π be an interpretation such that $\pi(r'', m'')(p) = \textbf{true}$ iff $(r'', m'') \in \mathcal{K}_j(r', m')$. Let $\mathcal{I} = (\mathcal{R}, \pi)$. Clearly, p is j-local. By assumption, $(\mathcal{I}, r, m) \models P_i p$. Thus, there exists some point $(r'', m'') \in \mathcal{K}_i(r, m)$ such that $(\mathcal{I}, r'', m'') \models p$. By definition, $(r'', m'') \in \mathcal{K}_j(r', m')$. Because $(r'', m'') \in \mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m')$, j maintains C-secrecy with respect to i in $\mathcal{R}$. □

Theorem 2: *Agent j maintains run-based secrecy with respect to agent i in system*

$\mathcal{R}$ iff, for every interpretation π, if ϕ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i \Diamond \phi$.

Proof: Suppose that j maintains run-based secrecy with respect to i. Let π be an interpretation and let ϕ be a j-local formula formula that is satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$. Choose a point (r, m). Because ϕ is satisfiable, there exists a point (r', m') such that $(\mathcal{I}, r', m') \models \phi$. Because j maintains run-based secrecy with respect to i, there exist a run r'' and times n and n' such that $r''_i(n) = r_i(m)$ and $r''_j(n') = r'_j(m')$. By j-locality, $(\mathcal{I}, r'', n') \models \phi$. It follows that $(\mathcal{I}, r'', n) \models \Diamond \phi$, and that $(\mathcal{I}, r, m) \models P_i \Diamond \phi$, as desired.

For the converse, given points (r, m) and (r', m'), let π be an interpretation such that $\pi(r'', m'')(p) = \textbf{true}$ iff $(r'', m'') \in \mathcal{K}_j(r', m')$. We must show that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap \mathcal{R}(\mathcal{K}_j(r', m')) \neq \emptyset$. Clearly p is j-local and satisfiable, so $(\mathcal{I}, r, m) \models P_i \Diamond p$. Thus, there exists a point $(r'', n) \in \mathcal{K}_i(r, m)$ such that $(\mathcal{I}, r'', n) \models \Diamond p$. By definition of p, there exists n' such that $(r'', n') \in \mathcal{K}_j(r', m')$. It follows that $r'' \in \mathcal{R}(\mathcal{K}_i(r, m)) \cap \mathcal{R}(\mathcal{K}_j(r', m'))$. □

A.3 Proofs for Section 3.2

Proposition 5: *If $(\mathcal{R}, \mathcal{PR})$ is a probability system such that $\mu_{r,m,i}(\{(r, m)\}) > 0$ for all points (r, m) and j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, then j also maintains total secrecy with respect to i in $\mathcal{R}$.*

Proof: Suppose that j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, and (r, m) and (r', m') are arbitrary points. Then (taking $(r'', m'') = (r', m')$ in the definition) we have $\mu_{r,m,i}(\mathcal{K}_j(r', m') \cap \mathcal{K}_i(r, m)) = \mu_{r',m',i}(\mathcal{K}_j(r', m') \cap \mathcal{K}_i(r', m'))$. But $(r', m') \in \mathcal{K}_j(r', m') \cap \mathcal{K}_i(r', m')$, so $\mu_{r',m',i}(\mathcal{K}_j(r', m') \cap \mathcal{K}_i(r', m')) \geq$

$\mu_{r',m',i}(\{(r',m')\}) > 0$, by assumption. Thus, $\mu_{r,m,i}(\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r,m)) > 0$, from which it follows that $\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r,m) \neq \emptyset$. □

The following result is proved by Gill, van der Laan, and Robins [28]; see also Grünwald and Halpern [33, Theorem 3.1]. (A more general version is stated and proved as Proposition 3.)

Lemma 1 *Suppose that μ is a probability on W, $X, Y \subseteq W$, $Y_1, Y_2, \ldots$ is a countable partition of $Y \subseteq W$, and $X, Y_1, Y_2, \ldots$ are all measurable. The following are equivalent:*

(a) $\mu(X \mid Y_i) = \mu(X \mid Y_j)$ *for all* Y_i, Y_j *such that* $\mu(Y_i) > 0$ *and* $\mu(Y_j) > 0$.

(b) $\mu(X \mid Y_i) = \mu(X \mid Y)$ *for all* Y_i *such that* $\mu(Y_i) > 0$, *so that* Y_i *is conditionally independent of* X *given* Y.

Proposition 6: *If $(\mathcal{R}, \mathcal{PR})$ is a probability system (resp., synchronous probability system) that satisfies the common prior assumption with prior probability μ_{cp}, the following are equivalent:*

(a) Agent j maintains probabilistic total (resp., synchronous) secrecy with respect to i.

(b) Agent i maintains probabilistic total (resp., synchronous) secrecy with respect to j.

(c) For all points (r,m) and (r',m'),

$$\mu_{cp}(\mathcal{K}_j(r',m') \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m'))$$

(resp., for all points (r,m) and (r',m),

$$\mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{PT}(m)),$$

where $\mathcal{PT}(m)$ is the set of points occurring at time m; that is, the events $\mathcal{K}_i(r,m)$ and $\mathcal{K}_j(r',m)$ are conditionally independent with respect to μ_{cp}, given that the time is m).

Proof: We prove the synchronous case here. The proof for total secrecy is almost identical and left to the reader. Recall that j maintains probabilistic synchronous secrecy with respect to i if, for all times m and all runs r, r', r'',

$$\mu_{r,m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r,m)) = \mu_{r',m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r',m)).$$

Because $(\mathcal{R}, \mathcal{PR})$ satisfies the common prior assumption with prior probability μ_{cp}, this requirement can be restated as

$$\mu_{cp}(\mathcal{K}_j(r'',m) \,|\, \mathcal{K}_i(r,m)) = \mu_{cp}(K_j(r'',m) \,|\, \mathcal{K}_i(r',m)).$$

By Lemma 1, taking $Y = \mathcal{PT}(m)$ and the Y_i's to be the i-information sets at time m, it follows that j maintains probabilistic synchronous secrecy with respect to i iff $\mathcal{K}_j(r'',m)$ is conditionally independent of $\mathcal{K}_i(r,m)$ conditional on $\mathcal{PT}(m)$ for all runs r and r''. By the symmetry of conditional independence, it immediately follows that this is true iff i maintains probabilistic synchronous secrecy with respect to j. □

Lemma 2 *If $\mathcal{R}$ is a system where agent i has perfect recall and Ω is an arbitrary set of i-information sets, then there exists a set $\Omega' \subseteq \Omega$ such that $\{R(\mathcal{K}) \,|\, \mathcal{K} \in \Omega'\}$ is a partition of $\bigcup_{\mathcal{K}\in\Omega} \mathcal{R}(\mathcal{K})$.*

Proof: Define a set $\mathcal{K} \in \Omega$ to be *dominated* by a set $\mathcal{K}' \in \Omega$ if $\mathcal{K} \neq \mathcal{K}'$ and there exists a run r and times $m' \langle m$ such that $(r,m) \in \mathcal{K}$ and $(r,m') \in \mathcal{K}'$. Let Ω' consist of the information sets in Ω that are not dominated by another set in Ω. Note

that if $r \in \cup_{\mathcal{K}\in\Omega}\mathcal{R}(\mathcal{K})$, then $r \in \mathcal{R}(\mathcal{K}')$ for some $\mathcal{K}' \in \Omega'$. To see this, consider the set $\Omega(\mathcal{K})$ consisting of $\mathcal{K}$ and all information sets in Ω that dominate $\mathcal{K}$. By perfect recall, i's local state sequence at each information set in $\Omega(\mathcal{K})$ is a (not necessarily strict) prefix of i's local state sequence in $\mathcal{K}$. Let $\mathcal{K}'$ be the information set in $\Omega(\mathcal{K})$ where i's local state sequence is shortest. It follows that $\mathcal{K}'$ is not dominated by another information set in $\Omega(\mathcal{K})$. Furthermore, if there exists an information set $\mathcal{K}'' \in \Omega - \Omega(\mathcal{K})$ that dominates $\mathcal{K}'$, then $\mathcal{K}''$ would dominate $\mathcal{K}$ as well, contradicting the construction of $\Omega(K)$. Therefore, $\mathcal{K}' \in \Omega'$ and $r \in \mathcal{K}'$. Thus $\bigcup_{\mathcal{K}\in\Omega'} \mathcal{R}(\mathcal{K}) = \bigcup_{\mathcal{K}\in\Omega} \mathcal{R}(\mathcal{K})$. Moreover, if $\mathcal{K}$ and $\mathcal{K}'$ are different sets in Ω', then $\mathcal{R}(\mathcal{K})$ and $\mathcal{R}(\mathcal{K}')$ must be disjoint, for otherwise one of $\mathcal{K}$ or $\mathcal{K}'$ would dominate the other. □

Proposition 7: *If $(\mathcal{R}, \mathcal{F}, \mu)$ is a run-based probability system that is either synchronous or one where agents i and j both have perfect recall, then the following are equivalent:*

(a) Agent j maintains run-based probabilistic secrecy with respect to i.

(b) Agent i maintains run-based probabilistic secrecy with respect to j.

(c) For all points (r, m) and (r', m'), $\mathcal{R}(\mathcal{K}_i(r, m))$ and $\mathcal{R}(\mathcal{K}_j(r', m'))$ are probabilistically independent with respect to μ.

Proof: First, note that if $\mathcal{R}$ is synchronous or if i has perfect recall, then there exists a collection Ω of i-information sets such that the set $\{R(\mathcal{K}) \,|\, \mathcal{K} \in \Omega\}$ is a partition of $\mathcal{R}$. In the case of perfect recall, this follows by Lemma 2 applied to the set of all information sets (whose union is clearly $\mathcal{R}$). With synchrony we can take Ω to consist of sets of the form $\mathcal{R}(\mathcal{K}_i(r, m))$, for some fixed time m.

Now, suppose j maintains run-based probabilistic secrecy with respect to i. By definition,

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r', m')))$$

for all points (r, m), (r', m'), and (r'', m''). In particular, for all $\mathcal{K}, \mathcal{K}' \in \Omega$, and all points (r', m'), $\mu(\mathcal{R}(\mathcal{K}_j(r', m') \,|\, \mathcal{R}(\mathcal{K}))) = \mu(\mathcal{R}(\mathcal{K}_j(r', m')) \,|\, \mathcal{R}(\mathcal{K}'))$. By Lemma 1 it follows that $\mu(\mathcal{R}(\mathcal{K}_j(r', m'')) \,|\, \mathcal{R}(\mathcal{K})) = \mu(\mathcal{R}(\mathcal{K}_j(r', m'')))$ for all information sets $\mathcal{K} \in \Omega$. But then it follows by secrecy that $\mu(\mathcal{R}(\mathcal{K}_j(r', m')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r', m'))$ for all i-information sets $\mathcal{R}(\mathcal{K}_i(r, m))$. Therefore $\mathcal{R}(\mathcal{K}_j(r', m'))$ and $\mathcal{R}(\mathcal{K}_i(r, m))$ are independent for all information sets $\mathcal{K}_j(r', m')$ and $\mathcal{K}_i(r, m)$. Thus secrecy implies independence, and this holds if we reverse the roles of i and j.

It is also clear that independence implies secrecy. For suppose that (c) holds. Then, for all points (r, m), (r', m'), and (r'', m''), we have

$$\begin{aligned} \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) &= \mu(\mathcal{R}(\mathcal{K}_j(r'', m''))) \\ &= \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r', m'))), \end{aligned}$$

so that j maintains run-based probabilistic secrecy with respect to i. Similarly, i maintains secrecy with respect to j. □

Proposition 8: *If $(\mathcal{R}, \mathcal{PR})$ is the standard system determined by the synchronous run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$ and agents i and j have perfect recall in $\mathcal{R}$, then agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$ iff j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$.*

Proof: Clearly if j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mu)$ and $(\mathcal{R}, \mathcal{PR})$ is the standard system determined by $(\mathcal{R}, \mu)$ then, at all times

m,

$$\begin{aligned}\mu_{r,m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r,m)) &= \mu(\mathcal{K}_j(r'',m) \,|\, \mathcal{K}_i(r,m)) \\ &= \mu(\mathcal{K}_j(r'',m) \,|\, \mathcal{K}_i(r',m)) \\ &= \mu_{r',m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r',m)),\end{aligned}$$

so j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$.

For the converse, suppose that j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$. We want to show that, for all points (r,m), (r',m'), and (r'',m''),

$$\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r',m'))). \tag{A.1}$$

We first show that, for all runs r and r'' and times m and m'',

$$\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m''))). \tag{A.2}$$

Since (A.2) also holds with r replaced by r', (A.1) easily follows from (A.2) and the assumption that j maintains probabilistic synchronous secrecy with respect to i.

To prove (A.2), we consider two cases: $m \leq m''$ and $m'' < m$. If $m \leq m''$ then, by synchrony and perfect recall, we can partition the runs in $\mathcal{R}(\mathcal{K}_i(r,m))$ according to i's local state at time m''. Let $\Omega = \{\mathcal{K}_i(r^*,m'') \,|\, r^* \in \mathcal{R}(\mathcal{K}_i(r,m))\}$. By perfect recall and synchrony, $\mathcal{R}(\mathcal{K}_i(r,m))$ is the disjoint union of the sets in Ω. Thus,

$$\begin{aligned}&\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ =\ &\textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \cap \mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ =\ &\textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K})) \cdot \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ =\ &\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m''))) \cdot \textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ =\ &\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m''))).\end{aligned}$$

The argument is similar if $m'' < m$. We now partition the runs in $\mathcal{R}(\mathcal{K}_i(r, m''))$ according to i's local state at time m and the runs in $\mathcal{R}(\mathcal{K}_j(r'', m''))$ according to j's local state at time m. Define

$$\Omega_i = \{\mathcal{K}_i(r^*, m) \,|\, r^* \in \mathcal{R}(\mathcal{K}_i(r, m'')\}.$$

and

$$\Omega_j = \{\mathcal{K}_j(r^*, m) \,|\, r^* \in \mathcal{R}(\mathcal{K}_j(r'', m''))\}.$$

We now have

$$\begin{aligned}
& \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m''))) \\
= \;& \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, \mathcal{R}(\mathcal{K}_i(r, m''))) \\
= \;& \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \sum_{\mathcal{K}_i \in \Omega_i} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, R(\mathcal{K}_i)) \cdot \mu(\mathcal{R}(\mathcal{K}_i) \,|\, \mathcal{R}(\mathcal{K}_i(r, m''))) \\
= \;& \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, R(\mathcal{K}_i)) \cdot \sum_{\mathcal{K}_i \in \Omega_i} \mu(\mathcal{R}(\mathcal{K}_i) \,|\, \mathcal{R}(\mathcal{K}_i(r, m''))) \\
= \;& \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) \\
= \;& \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))),
\end{aligned}$$

as needed. □

Theorem 3:

(a) If $(\mathcal{R}, \mathcal{PR})$ is a probabilistic system, then agent j maintains probabilistic total secrecy with respect to agent i iff, for every interpretation π and formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\phi) = \sigma$.

(b) If $(\mathcal{R}, \mathcal{PR})$ is a synchronous probabilistic system, then agent j maintains probabilistic synchronous secrecy with respect to agent i iff, for every interpretation π, time m, and formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\phi) = \sigma_m$ for all runs $r \in \mathcal{R}$.

Proof: We prove part (b) here. The proof of (a) is similar.

Suppose $\mathcal{R}$ is synchronous and that j maintains synchronous probabilistic secrecy with respect to i. Let π be an interpretation, m be an arbitrary time, and ϕ be a j-local formula in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$. Because ϕ is j-local, by Proposition 4, there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \phi$ iff $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Let $\Psi = \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Let $S = \{r' \in \mathcal{R} \,|\, (r', m) \in \Psi\}$, and let $\Omega(m) = \{\mathcal{K} \in \Omega \,|\, (r', m) \in \mathcal{K} \text{ for some } r' \in \mathcal{R}\}$. Since j maintains synchronous probabilistic secrecy with respect to i, for every element $\mathcal{K} \in \Omega(m)$, there is a constant $\sigma(\mathcal{K}, m)$ such that, for all runs $r \in \mathcal{R}$, $\mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma(\mathcal{K}, m)$. Let $\sigma_m = \sum_{\mathcal{K} \in \Omega(m)} \sigma(\mathcal{K}, m)$, and fix $r \in \mathcal{R}$. By synchrony, the set $\{\mathcal{R}(\mathcal{K}) \,|\, \mathcal{K} \in \Omega(m)\}$ partitions S, and

$$\mu(S \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sum_{\mathcal{K} \in \Omega(m)} \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma_m.$$

Because $\Psi \cap \mathcal{K}_i(r, m) = \mathcal{K}_i(r, m)(S)$, we have $\mu_{r,m,i}(\Psi) = \mu(S \,|\, \mathcal{R}(\mathcal{K}_i(r, m)))$, and it follows that $(\mathcal{I}, r, m) \models \Pr_i(\phi) = \sigma_m$, as desired.

For the converse, suppose that for every interpretation π and time m, if ϕ is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, then there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\phi) = \sigma_m$ for all runs $r \in \mathcal{R}$. Fix a time m. Suppose that $r, r', r'' \in \mathcal{R}$ and that π is an interpretation such that $\pi(r^*, n)(p) = \textbf{true}$ iff $(r^*, n) \in \mathcal{K}_j(r'', m)$. The proposition p is j-local, so there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(p) = \sigma_m$ and $(\mathcal{I}, r', m) \models \Pr_i(p) = \sigma_m$. It follows that

$$\mu_{r,m,i}(\mathcal{K}_j(r'', m)) = \sigma_m = \mu_{r',m,i}(\mathcal{K}_j(r'', m)),$$

as desired. □

Theorem 4: *If $(\mathcal{R}, \mathcal{PR})$ is a standard probability system where agent j has perfect recall, then agent j maintains run-based probabilistic secrecy with respect*

to agent i iff, for every interpretation π and every formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\Diamond\phi) = \sigma$.

Proof: Suppose that j maintains probabilistic secrecy with respect to agent i in $(\mathcal{R}, \mu)$. Given an interpretation π and a formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, by Proposition 4 there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \phi$ whenever $(r, m) \in \bigcup_{\mathcal{K}\in\Omega} \mathcal{K}$. Let $\Psi = \bigcup_{\mathcal{K}\in\Omega} \mathcal{R}(\mathcal{K})$. Note that $(\mathcal{I}, r, m) \models \Diamond\phi$ iff $r \in \mathcal{R}(\bigcup_{\mathcal{K}\in\Omega} \mathcal{K}) = \Psi$. By Lemma 2, there exists a set $\Omega' \subseteq \Omega$ such that $\{\mathcal{R}(\mathcal{K}) : \mathcal{K} \in \Omega\}$ is a partition of Ψ. By probabilistic secrecy, for each $\mathcal{K} \in \Omega'$, there exists a constant $\sigma_{\mathcal{K}}$ such that

$$\mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma_{\mathcal{K}}$$

for all points (r, m). Let $\sigma = \sum_{\mathcal{K}\in\Omega'} \sigma_{\mathcal{K}}$. Because $\{\mathcal{R}(\mathcal{K}) \,|\, \mathcal{K} \in \Omega'\}$ is a partition of Ψ, for all points (r, m),

$$\mu(\Psi \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma.$$

Because $\mu_{r,m,i}(\mathcal{K}_i(r, m)(\Psi)) = \mu(\Psi \,|\, \mathcal{R}(\mathcal{K}_i(r, m)))$, it follows that $\mathcal{I} \models \Pr_i(\Diamond\phi) = \sigma$.

For the converse, suppose that for every interpretation π and formula ϕ that is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\Diamond\phi) = \sigma$. Given points (r, m), (r', m'), and (r'', m''), let π be an interpretation such that $\pi(r^*, n)(p) = \mathbf{true}$ iff $(r^*, n) \in \mathcal{K}_j(r'', m'')$. The proposition p is j-local, so $\mathcal{I} \models \Pr_i(\Diamond p) = \sigma$. It follows that

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu_{r,m,i}(\mathcal{K}_i(r, m)(\mathcal{R}(\mathcal{K}_j(r'', m'')))) = \sigma,$$

and the same holds if we replace (r, m) with (r', m'), so

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r', m'))).$$

This gives us probabilistic secrecy. □

Theorem 5: *Let $(\mathcal{R}, \mathcal{D}, \Delta)$ be the adversarial probability system determined by INIT and suppose that $\mathcal{R}$ is either synchronous or a system where i has perfect recall. Agent i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$ iff agent i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$.*

Proof: For the forward direction, we want to show that i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$. Suppose that $\mu \in \mathcal{M}_i^{INIT}(\Delta)$. The information function f_{i^-} maps an i^--information set to the choices made by the agents other than i. Let an i^--*choice set* be a set of runs of the form $\cap_{j \neq i} D_{y_j}$. We must show that for arbitrary points (r, m) and (r', m') and i^--choice sets D_{i^-}, we have

$$\mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r', m'))). \tag{A.3}$$

Since, by assumption, i's choice is encoded i's local state, there exists a unique y_i such that $\mathcal{R}(\mathcal{K}_i(r, m)) \subseteq D_{y_i}$. Since i obtains no evidence for the initial choice, we have that for all i^--choice sets D_{i^-} and D'_{i^-},

$$\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r, m))) = \mu_{D_{y_i} \cap D'_{i^-}}(\mathcal{R}(\mathcal{K}_i(r, m))). \tag{A.4}$$

Thus, whenever $\mu(D_{y_i} \cap D_{i^-}) > 0$ and $\mu(D_{y_i} \cap D'_{i^-}) > 0$, we have

$$\begin{aligned} \mu(\mathcal{R}(\mathcal{K}_i(r, m)) \mid D_{y_i} \cap D_{i^-}) &= \mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r, m))) \\ &= \mu_{D_{y_i} \cap D'_{i^-}}(\mathcal{R}(\mathcal{K}_i(r, m))) \\ &= \mu(\mathcal{R}(\mathcal{K}_i(r, m)) \mid D_{y_i} \cap D'_{i^-}). \end{aligned}$$

It now follows by Lemma 1 that $\mathcal{R}(\mathcal{K}_i(r, m))$ is conditionally independent of every i^--choice set given D_{y_i}. (Though Lemma 1 actually shows only that $\mathcal{R}(\mathcal{K}_i(r, m))$

is conditionally independent of every i^- choice set D_{i^-} such that $\mu(D_{i^-} \cap D_{y_i}) > 0$, conditional independence is immediate if $\mu(D_{i^-} \cap D_{y_i}) = 0$) Thus, for any i^--choice set D_{i^-}, we have

$$\mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i}) = \mu(D_{i^-} \mid D_{y_i}) = \mu(D_{i^-}),$$

where the last equality follows because we have assumed that i's choice is independent of the choices made by other agents. Similarly, $\mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r',m'))) = \mu(D_{i^-})$, so (A.3) follows, and i^- does indeed maintain generalized run-based probabilistic f_{i^-}-secrecy with respect to i.

For the converse, suppose that i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i. Thus, for all points (r,m), i^--choice sets D_{i^-}, and measures $\mu \in \mathcal{M}_i^{INIT}(\Delta)$, we have (A.3). Given two i^--choice sets D_{i^-} and D'_{i^-} and an i-information set $\mathcal{K}_i(r,m)$ such that $\mathcal{R}(\mathcal{K}_i(r,m)) \subseteq D_{y_i}$, we want to show (A.4). To do so we first show that there exists a measure $\mu \in \mathcal{M}_i^{INIT}(\Delta)$ that places positive probability on all the cells. (We will make use of this particular measure for the duration of the proof.) Our strategy is to take a countable linear combination of the cell-specific probability measures, such that the set of runs in each cell is assigned positive probability by μ. Let $y_{i1}, y_{i2}, \ldots$ be a countable enumeration of $INIT_i$, and let $D_1, D_2, \ldots$ be a countable enumeration of the possible i^--choice sets. Define the function μ such that for $U \in \mathcal{F}$,

$$\mu(U) = \sum_{j \geq 1, k \geq 1} \frac{\mu_{D_{y_{ij}} \cap D_k}(U \cap D_{y_{ij}} \cap D_k)}{2^{jk}}.$$

It is straightforward to check that $\mu \in \mathcal{M}_i^{INIT}(\Delta)$ and that it places a positive probability on all the cells in $\mathcal{D}$. Furthermore, we have $\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-})$, and the same holds if we replace D_{i^-} with D'_{i^-}.

Given an i-information set $\mathcal{K}_i(r,m)$, let y_i be the initial choice for i such that $\mathcal{R}(\mathcal{K}_i(r,m)) \subseteq D_{y_i}$. For all i^- choice sets D_{i^-}, we have

$$\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-}).$$

Thus, to prove (A.4), it suffices to show that

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D'_{i^-}).$$

Standard probabilistic manipulations show that

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-}) \cdot \mu(D_{y_i} \mid D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i} \mid D_{i^-}); \quad \text{(A.5)}$$

a similar equation holds if we replace D_{i^-} by D'_{i^-}. Since either $\mathcal{R}$ is synchronous or i has perfect recall in $\mathcal{R}$, there exists a set Ω of i-information sets such that $\{\mathcal{R}(\mathcal{K}) : \mathcal{K} \in \Omega\}$ partitions $\mathcal{R}$. By Lemma 1 and (A.3), it follows that i^--choice sets are independent of the i-information sets in Ω. Applying (A.3) again, it follows that i^--choice sets are independent of *all* i-information sets. Thus, $\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i} \mid D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)))$. Since D_{i^-} and D_{y_i} are independent by assumption, it follows that $\mu(D_{y_i} \mid D_{i^-}) = \mu(D_{y_i})$. Thus, (A.5) reduces to

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-}) \cdot \mu(D_{y_i}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m))).$$

The same is true for D'_{i^-}, so because $\mu(D_{y_i}) > 0$ it follows that $\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid D_{y_i} \cap D'_{i^-})$. (A.4) is now immediate. □

A.4 Generalizing from probability to plausibility

In this section we give the details of the plausibilistic results presented in Section 3.3. All those results correspond to probabilistic results from the previous section;

in many cases the proofs are almost identical. For brevity we focus here on the nontrivial subtleties that arise in the plausibilistic case.

To show that Proposition 8 generalizes to run-based plausibility systems is straightforward. We simply replace all occurrences of multiplication and addition in the proof of Proposition 8 with $\otimes$ and $\oplus$; all the resulting equations hold by the properties of cacps's.

To define analogues of Theorems 3 and 4, we need a language that allows statements of the form $\mathrm{Pl}_i(\phi) = c$, where c is a constant that is interpreted as a plausibility value. Once we do this, the proofs of these results transfer to the plausibilistic setting with almost no change. We omit the straightforward details.

To prove Propositions 6 and 7, we first prove two results that generalize Lemma 1. To do so, we need the following definition, taken from [36]. Define a cacps to be *acceptable* if $U \in \mathcal{F}'$ and $\mathrm{Pl}(V \mid U) \neq \bot$ implies that $V \cap U \in \mathcal{F}'$. To understand the intuition behind this definition, consider the special case where $U = W$. Since $W \in \mathcal{F}'$ (this follows from the fact that $\mathcal{F}'$ is a nonempty and is closed under supersets in $\mathcal{F}$), we get that if $\mathrm{Pl}(V) \neq \bot$, then $V \in \mathcal{F}'$. This is an analogue of the situation in probability, where we can always condition on a set of nonzero measure.

Lemma 3 *Let $(W, \mathcal{F}, \mathcal{F}', Pl)$ be an acceptable cacps. Suppose that $Y_1, Y_2, \ldots$ is a partition of $Y \in \mathcal{F}'$, and that $Y_i \in \mathcal{F}$ for $i = 1, 2, 3, \ldots$. For all $X \in \mathcal{F}$, the following are equivalent:*

(a) $Pl(X \mid Y_i) = Pl(X \mid Y_j)$ for all $Y_i, Y_j \in \mathcal{F}'$.

(b) $Pl(X \mid Y_i) = Pl(X \mid Y)$ for all $Y_i \in \mathcal{F}'$.

Proof: Clearly (b) implies (a). To see that (a) implies (b), first note that since we

are dealing with an acceptable cacps, if $Y_j \notin \mathcal{F}'$, then $\mathrm{Pl}(Y_j \mid Y) = \bot$ and hence, for all X, $\mathrm{Pl}(X \cap Y_j \mid Y) = \bot$. Given $Y_i \in \mathcal{F}'$, it follows that

$$\begin{aligned}
\mathrm{Pl}(X \mid Y) &= \oplus_{\{j:Y_j \in \mathcal{F}'\}} \mathrm{Pl}(X \cap Y_j \mid Y) \\
&= \oplus_{\{j:Y_j \in \mathcal{F}'\}} (\mathrm{Pl}(X \mid Y_j) \otimes \mathrm{Pl}(Y_j \mid Y)) \\
&= \oplus_{\{j:Y_j \in \mathcal{F}'\}} (\mathrm{Pl}(X \mid Y_i) \otimes \mathrm{Pl}(Y_j \mid Y)) \\
&= \mathrm{Pl}(X \mid Y_i) \otimes (\oplus_{\{j:Y_j \in \mathcal{F}'\}} \mathrm{Pl}(Y_j \mid Y)) \\
&= \mathrm{Pl}(X \mid Y_i),
\end{aligned}$$

as needed. □

In the probabilistic setting, if either part (a) or (b) of Proposition 1 holds, we are able to conclude that Y_i is conditionally independent of X given Y. By the symmetry of independence in the probabilistic setting, we can conclude that X is also conditionally independent of Y_i given Y, that is, that $\Pr(Y_i \mid X \cap Y) = \Pr(Y_i \mid Y)$. In the plausibilistic setting, independence is not symmetric in general unless we make an additional assumption, namely that $\otimes$ is symmetric. We say that a cacps is *commutative* if its $\otimes$ operator is commutative.

Lemma 4 *Suppose that $(W, \mathcal{F}, \mathcal{F}', Pl)$ is a commutative acceptable cacps; $Y_1, Y_2, \ldots$ is a partition of $Y \in \mathcal{F}'$; $X \in \mathcal{F}'$, $X \subseteq Y$, and $Pl(X \mid Y) \neq \bot$; and for all $Y_i, Y_j \in \mathcal{F}'$, $Pl(X \mid Y_i) = Pl(X \mid Y_j)$. Then, for all $Y_i \in \mathcal{F}$, $Pl(Y_i \mid X) = Pl(Y_i \mid Y)$.*

Proof: First, suppose that $Y_i \in \mathcal{F}'$. By Lemma 3, we have that $\mathrm{Pl}(X \mid Y_i) = \mathrm{Pl}(X \mid Y)$. Since $Y_i \cap Y = Y_i$ and $\otimes$ is commutative, we have

$$\begin{aligned}
\mathrm{Pl}(X \cap Y_i \mid Y) &= \mathrm{Pl}(X \mid Y_i) \otimes \mathrm{Pl}(Y_i \mid Y) \\
&= \mathrm{Pl}(X \mid Y) \otimes \mathrm{Pl}(Y_i \mid Y) \\
&= \mathrm{Pl}(Y_i \mid Y) \otimes \mathrm{Pl}(X \mid Y).
\end{aligned}$$

Similarly, since $X \subseteq Y$, we have

$$\mathrm{Pl}(X \cap Y_i \,|\, Y) = \mathrm{Pl}(Y_i \,|\, X) \otimes \mathrm{Pl}(X \,|\, Y).$$

Thus, $\mathrm{Pl}(Y_i \,|\, Y) \otimes \mathrm{Pl}(X \,|\, Y) = \mathrm{Pl}(Y_i \,|\, X) \otimes \mathrm{Pl}(X \,|\, Y)$. Since $\mathrm{Pl}(X \,|\, Y) \neq \bot$ by assumption, it follows from the definition of a cacps that $\mathrm{Pl}(Y_i \,|\, Y) = \mathrm{Pl}(Y_i \,|\, X)$.

If $Y_i \notin \mathcal{F}'$ but $Y_i \in \mathcal{F}$, then $Y_i \cap X \notin \mathcal{F}'$ (since $\mathcal{F}'$ is closed under supersets in $\mathcal{F}$). Since we are working in an acceptable cacps, $\mathrm{Pl}(Y_i \,|\, Y) = \bot$ and $\mathrm{Pl}(Y_i \,|\, X) = \bot$, so again $\mathrm{Pl}(Y_i \,|\, Y) = \mathrm{Pl}(Y_i \,|\, X)$. □

With these results, plausibilistic versions of Propositions 6 and 7 can be proved with only minor changes to the proof in the probabilistic case, provided we make the additional assumptions stated in the main text. We replace the use of Lemma 1 by Lemma 3. The appeal to the symmetry of conditional independence is replaced by an appeal to Lemma 4. However, to use this lemma, we need to assume that $\otimes$ is commutative and that for all points (r, m),

- $\mathrm{Pl}_{cp}(\mathcal{K}_i(r,m) \,|\, \mathcal{PT}(\mathcal{R})) \neq \bot$ and $\mathrm{Pl}_{cp}(\mathcal{K}_j(r,m) \,|\, \mathcal{PT}(\mathcal{R})) \neq \bot$ (in the proof of total secrecy in the generalization of Proposition 6);
- $\mathrm{Pl}_{cp}(\mathcal{K}_i(r,m) \,|\, \mathcal{PT}(m)) \neq \bot$ and $\mathrm{Pl}_{cp}(\mathcal{K}_j(r,m) \,|\, \mathcal{PT}(m)) \neq \bot$ (in the the proof of synchronous secrecy in the generalization of Proposition 6); and
- $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, \mathcal{R}) \neq \bot$ and $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_j(r,m)) \,|\, \mathcal{R}) \neq \bot$ (in the generalization of Proposition 7).

(We do not have to assume that the relevant cacps's are acceptable for these propositions; it is enough that they are commutative. We used acceptability in the proof of Lemma 3 to show argue that if a set Y_i is not in $\mathcal{F}'$, then $\mathrm{Pl}(Y_i) = \bot$.

Here, the sets Y_i are of the form $\mathcal{R}(\mathcal{K}_i(r,m))$, and our other assumptions guarantee that they are in $\mathcal{F}'$.)

Turning to the generalization of Theorem 5, the first step is to define an *adversarial plausibility system*. The definition is completely analogous to that of that of an adversarial probability system, except that now the set Δ consists of the acceptable conditional plausibility spaces $(D, \mathcal{F}_D, \mathcal{F}'_D, \mathrm{Pl}_D)$, for each cell $D \in \mathcal{D}$. Again, we assume that $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \in \mathcal{F}_D$ and that, if $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \neq \emptyset$, then $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \in \mathcal{F}'_D$ and $\mathrm{Pl}_D(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D) \neq \bot$. We say that an agent i *obtains no plausibilistic evidence for the initial choice* in $(\mathcal{R}, \mathcal{D}, \Delta)$ if for all $D, D' \in \mathcal{D}$ and all points (r,m) such that $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \neq \emptyset$ and $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D' \neq \emptyset$, we have

$$\mathrm{Pl}_D(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D) = \mathrm{Pl}_{D'}(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D').$$

Suppose that $\mathcal{D}$ is determined by *INIT* (as in the probabilistic case), and that the conditional plausibility spaces of Δ are all defined with respect to the same domain $\mathbf{D}$ of plausibility values and with the same operations $\oplus$ and $\otimes$, where $\otimes$ is commutative. Let $\mathcal{F}_{\mathcal{D}}$ be the σ-algebra generated by $\cup_{D \in \mathcal{D}} \mathcal{F}_D$. Let $\mathcal{M}_i^{INIT,\mathrm{Pl}}(\Delta)$ consist of all the acceptable plausibility spaces $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mathcal{F}', \mathrm{Pl})$ such that

- $\mathcal{F}'$ is a nonempty subset of $\mathcal{F}_{\mathcal{D}}$ that is closed under supersets;
- if $A \in \mathcal{F}_D$ and $B \in \mathcal{F}' \cap \mathcal{F}'_D$, then $\mathrm{Pl}(A \mid B) = \mathrm{Pl}_D(A \mid B)$;
- for all agents i and points (r,m), there exists a cell D such that $\mathrm{Pl}(D) \neq \bot$ and $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \neq \emptyset$; and
- $\mathrm{Pl}(D_{(y_1,\ldots,y_n)}) = \mathrm{Pl}(D_{y_i}) \otimes \mathrm{Pl}(\cap_{j \neq i} D_{y_j})$.

We can now state and prove the plausibilistic analogue of Theorem 5.

Theorem 12 *Let $(\mathcal{R}, \mathcal{D}, \Delta)$ be the adversarial plausibility system determined by INIT and suppose that $\mathcal{R}$ is either synchronous or a system where i has perfect recall. Agent i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$ iff agent i^- maintains generalized run-based plausibilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT,Pl}(\Delta))$.*

Proof: The proof is basically the same as that of Theorem 5, but some new subtleties arise because we are dealing with plausibility. For the forward direction, we want to show that i^- maintains generalized run-based plausibilistic f_{i^-}-secrecy under the assumption that i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$. Much as in the proof of Theorem 5, we can show that $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-}) = \mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D'_{i^-})$ if $D_{i^-} \cap D_{y_i} \in \mathcal{F}'$ and $D'_{i^-} \cap D_{y_i} \in \mathcal{F}'$. Continuing in the spirit of that proof, we now want to show that $\mathrm{Pl}(D_{i^-} \,|\, \mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i}) = \mathrm{Pl}(D_{i^-} \,|\, D_{y_i}) = \mathrm{Pl}(D_{i^-})$. For the second equality, note that, by assumption, $\mathrm{Pl}(D_{i^-} \cap D_{y_i}) = \mathrm{Pl}(D_{i^-}) \otimes \mathrm{Pl}(D_{y_i})$. Since the properties of acceptable conditional plausibility spaces guarantee that $\mathrm{Pl}(D_{i^-} \,|\, D_{y_i}) \otimes \mathrm{Pl}(D_{y_i}) = \mathrm{Pl}(D_{i^-} \cap D_{y_i})$, it follows that $\mathrm{Pl}(D_{i^-} \,|\, D_{y_i}) \otimes \mathrm{Pl}(D_{y_i}) = \mathrm{Pl}(D_{i^-}) \otimes \mathrm{Pl}(D_{y_i})$. Since $\mathrm{Pl}(D_{y_i}) \neq \bot$, $\mathrm{Pl}(D_{i^-} \,|\, D_{y_i}) = \mathrm{Pl}(D_{i^-})$.

To prove the first equality, we want to apply Lemma 4. To do so, we must first show that $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i}) \neq \bot$. To see that this holds, recall that by assumption there exists a cell D such that $\mathrm{Pl}(D) \neq \bot$, $\mathrm{Pl}_D(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D) \neq \bot$, and $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \in \mathcal{F}'$. Since $\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \neq \emptyset$, we must have that $D \subseteq D_{y_i}$. Indeed, we must have $D = \widehat{D}_{i^-} \cap D_{y_i}$ for some i^--choice set $\widehat{D}_{i^-}$. Thus, we have

$$\begin{aligned} \mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i}) &\geq \mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D \,|\, D_{y_i}) \\ &= \mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D) \otimes \mathrm{Pl}(\widehat{D}_{i^-} \,|\, D_{y_i}) \end{aligned}$$

$$= \quad \mathrm{Pl}_D(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D) \otimes \mathrm{Pl}(\widehat{D}_{i^-}).$$

By assumption $\mathrm{Pl}_D(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D) \neq \bot$. Since $\mathrm{Pl}(D) \neq \bot$ and $D \subseteq \widehat{D}_{i^-}$, it follows that $\mathrm{Pl}(\widehat{D}_{i^-}) \neq \bot$. Thus, $\mathrm{Pl}(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i}) \neq \bot$.

For the converse, we must construct an acceptable measure Pl and a set $\mathcal{F}'$ such that $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mathcal{F}', \mathrm{Pl}) \in \mathcal{M}_i^{INIT,\mathrm{Pl}}(\Delta)$. We take $\mathcal{F}'$ to consist of the sets U such that $U \cap D \in \mathcal{F}'_D$ for some cell D. For Pl, we start by taking some arbitrary total ordering $\prec$ of the cells in $\mathcal{D}$. Given $V \in \mathcal{F}$ and $U \in \mathcal{F}'$, let $\mathrm{Pl}(V \,|\, U) = \mathrm{Pl}_D(V \cap D \,|\, U \cap D)$ where D is the highest-ranked cell such that $U \cap D \in \mathcal{F}_D$. By construction, Pl behaves identically to the cell-specific measures when we condition on subsets of cells. It is easy to check that for all $y_i \in INIT_i$ and i^--choice sets D_{i^-}, we have $\mathrm{Pl}(D_{y_i} \cap D_{i^-}) = \top$, $\mathrm{Pl}(D_{y_i}) = \top$, and $\mathrm{Pl}(D_{i^-}) = \top$. The independence of the choices made by i and i^- follows immediately.

To see that the measure satisfies the conditioning axiom (in the definition of a cacps), suppose that $U_1, U_2, U_3 \in \mathcal{F}$ and $U_2 \cap U_3 \in \mathcal{F}'$. We must show that $\mathrm{Pl}(U_1 \cap U_2 \,|\, U_3) = \mathrm{Pl}(U_1 \,|\, U_2 \cap U_3) \otimes \mathrm{Pl}(U_2 \,|\, U_3)$. There are two cases. If the highest-ranked cell that intersects U_3 (call it D) also intersects U_2, then all three terms in the equality are determined by Pl_D, and the equality follows by applying the conditioning axiom to Pl_D with $U_1 \cap D, U_2 \cap D$, and $U_3 \cap D$. If the highest-ranked cell D that intersects U_3 does not intersect U_2, then the first and third terms in the equality are both determined by Pl_D and must be $\bot$ because $U_2 \cap D = \emptyset$.

Finally, the measure Pl is acceptable (as required) because the underlying cell-specific measures are acceptable.

The remainder of the proof is a relatively straightforward extension of the probabilistic case. That i^--choice sets are independent of i-information sets follows from Lemma 4, using the facts that agent i^- maintains generalized run-based

plausibilistic f_{i^-}-secrecy, cells (and thus i^--choice sets) have non-$\perp$ plausibility by construction, and all information sets are in $\mathcal{F}'$. $\square$

Appendix B

Proofs for Chapter 5

Proposition 10: *If a synchronous trace system Σ satisfies separability (resp., generalized noninterference), then H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

Proof: We prove the result for separability. The proof for generalized noninterference is similar and left to the reader. Suppose that Σ satisfies separability. Let r^t and $r^{t'}$ be runs in $\mathcal{R}(\Sigma)$. We want to show that, for all times m, we have $\mathcal{K}_L(r^t, m) \cap \mathcal{K}_H(r^{t'}, m) \neq \emptyset$. Since Σ satisfies separability, there exists a trace $t'' \in \Sigma$ such that $t'' \restriction L = t \restriction L$ and $t'' \restriction H = t' \restriction H$. It follows immediately that $t''_m \restriction L = t_m \restriction L$ and $t''_m \restriction H = t'_m \restriction H$. Thus, $(r^{t''}, m) \in \mathcal{K}_L(r^t, m) \cap \mathcal{K}_H(r^{t'}, m)$. □

Proposition 11: *A limit-closed synchronous trace system Σ satisfies separability (resp. generalized noninterference) iff H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

Proof: We give the argument for separability here; the argument for generalized noninterference is similar. The forward direction follows from Proposition 10. For the converse, suppose that H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$. Given $t, t' \in \Sigma$, let t'' be the trace such that $t'' \restriction L = t \restriction L$ and $t'' \restriction H = t' \restriction H$. We must show that $t'' \in \Sigma$. Since H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$, for all m, there exists a run $r^m \in \mathcal{R}(\Sigma)$ such that $r^m_L(m) = r^t_L(m)$ and $r^m_H(m) = r^{t'}_H(m)$. Thus, for all m, there exists a trace $t^m \in \Sigma$ such that $t^m_m \restriction L = t_m \restriction L$ and $t^m_m \restriction H = t'_m \restriction H$. It follows that $t''_m = t^m_m$ for all m. Since $t^m \in \Sigma$ for all m, it follows by limit closure that $t'' \in \Sigma$, as desired. □

Proposition 12: *If Σ is an asynchronous trace system that satisfies asynchronous separability (resp. asynchronous generalized noninterference), then H maintains total secrecy (resp. total f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

Proof: Suppose that Σ satisfies asynchronous separability, and let (r, m) and (r', m') be arbitrary points. By the construction of $\mathcal{R}(\Sigma)$, there exist traces $t, t' \in T$ such that $r_L(m) = t \restriction L$ and $r_H(m) = t' \restriction H$. Let t'' be an interleaving of $t \restriction L$ and $t' \restriction H$. Since Σ satisfies asynchronous separability, $t'' \in \Sigma$. Let T'' be a run-like set of traces that contains t''. (Such a set must exist because Σ is closed under trace prefixes.) By definition, $r^{T''} \in \mathcal{R}(\Sigma)$. Taking m to be the length of t'', it follows that $r''_L(m'') = r_L(m)$ and $r''_H(m'') = r'_H(m')$. Thus, H maintains total secrecy with respect to L.

The proof for asynchronous generalized noninterference (and total f_{hi}-secrecy) is analogous. □

Proposition 13: *If Σ is an asynchronous trace system that is closed under interleavings, then Σ satisfies asynchronous separability iff H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

Proof: We have already established the forward direction. For the converse, suppose that H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, and that Σ is closed under interleavings. Given $t, t' \in \Sigma$, there exist points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R}(\Sigma))$ such that $r_L(m) = t \restriction L$ and $r'_H(m') = t' \restriction H$. Since H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, there exists a point (r'', m'') such that $r''_L(m'') = r_L(m)$ and $r''_H(m'') = r'_H(m')$. By the construction of $\mathcal{R}(\Sigma)$, there exists a run-like set T of traces such that $r'' = r^T$. Taking t'' to be the trace of length m'' in T, it follows that $t'' \restriction L = t \restriction L$ and $t'' \restriction H = t' \restriction H$. Because Σ is closed under

interleavings, $t'' \in \Sigma$ as required. □

Theorem 7: *If $\mathcal{I} = (\mathcal{R}, \pi)$ is compatible with P, then P is strongly anonymous on the alphabet A if and only if for every agent $i \in I_A$, the action a performed by i is anonymous up to I_A with respect to o in $\mathcal{I}$.*

Proof: Suppose that P is strongly anonymous on the alphabet A and that $i \in I_A$. Given a point (r, m), suppose that $(\mathcal{I}, r, m) \models \theta(i, a)$, so that the event $i.a$ appears in $r_e(n)$ for some $n \geq m$. We must show that $(\mathcal{I}, r, m) \models P_o[\theta(i', a)]$ for every $i' \in I_A$, that is, that a is anonymous up to I_A with respect to o. For any $i' \in I_A$, this requires showing that there exists a point (r', m') such that $r_o(m) = r'_o(m')$, and $r'_o(n')$ includes $i'.a$, for some $n' \geq m'$. Because $\mathcal{R}$ is compatible with P, there exists $t \in P$ such that $t = r_e(n)$ and $i.a$ appears in t. Let t' be the trace identical to t except that $i.a$ is replaced by $i'.a$. Because P is strongly anonymous on A, $P = f_A^{-1}(f_A(P))$, and $t' \in P$. By compatibility, there exists a run r' such that $r'_e(n) = t'$ and $r'_o(n) = f_A(t')$. By construction, $f_A(t) = f_A(t')$, so $r_o(n) = r'_o(n)$. Because the length-m trace prefixes of $f_A(t)$ and $f_A(t')$ are the same, it follows that $r_o(m) = r'_o(m)$. Because $(\mathcal{I}, r', m) \models \theta(i', a)$, $(\mathcal{I}, r, m) \models P_o[\theta(i', a)]$ as required.

Conversely, suppose that for every agent $i \in I_A$, the action a performed by i is anonymous up to I_A with respect to o in $\mathcal{I}$. We must show that P is strongly anonymous. It is clear that $P \subseteq f_A^{-1}(f_A(P))$, so we must show only that $P \supseteq f_A^{-1}(f_A(P))$. So suppose that $t \in f_A^{-1}(f_A(P))$. If no event $i.a$ appears in t, for any $i \in I_A$, then $t \in P$ trivially. Otherwise, some $i.a.$ does appear. Because $t \in f_A^{-1}(f_A(P))$, there exists a trace $t' \in P$ that is identical to t except that $i'.a$ replaces $i.a$, for some other $i' \in I_A$. Because $\mathcal{R}$ is compatible with P, there exists a run $r' \in R$ such that $r'_o(m) = f_A(t')$ and $r'_e(m) = t'$ (where $m = |t'|$). Clearly $(\mathcal{I}, r', m) \models \theta(i', a)$ so, by anonymity, $(\mathcal{I}, r', m) \models P_o[\theta(i, a)]$, and there exists a

run r such that $r_o(m) = r'_o(m)$ and $(\mathcal{I}, r, m) \models \theta(i, a)$. Because the action a can be performed at most once, the trace $r_e(m)$ must be equal to t. By compatibility, $t \in P$ as required. □

Theorem 8: *Suppose that* $(\mathcal{I}, r, m) \models \theta(i, a)$ *exactly if* $f_{(r,m)}(a) = i$. *Then action* a *is anonymous up to* I_A *with respect to* o *for each agent* $i \in I_A$ *if and only if at all points* (r, m) *such that* $f_{(r,m)}(a) \in I_A$, f *is* I_A*-value opaque with respect to* o.

Proof: Suppose that f is I_A-value opaque, and let $i \in I_A$ be given. If $(\mathcal{I}, r, m) \models \theta(i, a)$, then $f_{(r,m)}(a) = i$. We must show that, for all $i' \in I_A$, $(\mathcal{I}, r, m) \models P_o[\theta(i', a)]$. Because f is I_A-value opaque at (r, m), there exists a point (r', m') such that $r'_o(m') = r_o(m)$ and $f_{(r',m')}(a) = i'$. Because $(\mathcal{I}, r', m') \models \theta(i', a)$, $(\mathcal{I}, r, m) \models P_o[\theta(i', a)]$.

Conversely, suppose that for each agent $i \in I_A$, a is anonymous up to I_A with respect to o. Let (r, m) be given such that $f_{(r,m)}(a) \in I_A$, and let that $i = f_{(r,m)}(a)$. It follows that $(\mathcal{I}, r, m) \models \theta(i, a)$. For any $i' \in I_A$, $(\mathcal{I}, r, m) \models P_o[\theta(i', a)]$, by anonymity. Thus there exists a point (r', m') such that $r'_o(m') = r_o(m)$ and $(\mathcal{I}, r', m') \models \theta(i', a)$. It follows that $f_{(r',m')}(a) = i'$, and that f is I_A-value opaque.

□

Appendix C

Proof Sketch for Theorem 11

For the proof of Theorem 11, we treat the proof for nonprobabilistic noninterference—that is, the proof of parts (a) and (b)—separately from the proof of probabilistic noninterference. This is technically unnecessary, because the necessary lemmas for the probabilistic proof are generalizations of the lemmas for the nonprobabilistic proof. We take this approach so that we can prove the nonprobabilistic lemmas, which are simpler to understand, without the overhead of probability trees and probability distributions. The nonprobabilistic lemmas are proven in Section C.1, whereas the probabilistic lemmas and Theorem 11 are proven in Section C.2.

All of the results in this section assume the existence of a single variable typing Γ. When convenient, we avoid specifying Γ and assume that the typing is given.

We heavily overload the symbol $\sim_L$ to represent low equivalence relations. We write $\sigma \sim_L \sigma'$ to denote that states σ and σ' are low-equivalent with respect to Γ, that is, if $\sigma(x) = \sigma'(x)$ whenever $\Gamma(x) = L$. Refiners $\psi, \psi' \in \mathbf{Ref}$ are low-equivalent, written $\psi \sim_L \psi'$, if $\psi(L) = \psi'(L)$. Similarly, joint strategies $\omega, \omega' \in \mathbf{Strat}$ are low-equivalent, written $\omega \sim_L \omega'$ if $\omega(L) = \omega'(L)$. Traces t and t' are low-equivalent, written $t \sim_L t'$, if $t \restriction L = t' \restriction L$.

The low-equivalence relation on well-typed commands, denoted $c \sim_L c'$, is defined by the following rules:

(a) $c \sim_L c$ for all commands c;

(b) if $\Gamma \vdash c_1 : H\ cmd$ and $\Gamma \vdash c_2 : H\ cmd$, then $c_1 \sim_L c_2$;

(c) if $\Gamma \vdash c_1 : H\ cmd$ and $\Gamma \vdash c_2 : H\ cmd$, then $c_1; c \sim_L c_2; c$ for all commands

c; and

(d) if $\Gamma \vdash c_H : H\ cmd$, then $c_H; c \sim_L c$ and $c \sim_L c_H; c$ for all commands c.

Property 1 *The relation $\sim_L$ is an equivalence relation on well-typed commands.*

Proof: Reflexivity is immediate by rule (a), and symmetry follows because the rules themselves are symmetric. Transitivity follows by a straightforward analysis of each pair of rules. □

Two configurations $m = (c, \sigma, \psi, t, \omega)$ and $m' = (c', \sigma', \psi', t', \omega')$ are low-equivalent, written $m \sim_L m'$, if $c \sim_L c'$, $\sigma \sim_L \sigma'$, $\psi \sim_L \psi'$, $t \sim_L t'$, and $\omega \sim_L \omega'$.

C.1 Nonprobabilistic proof details

The following lemma, an analogue of the "Simple Security" lemma of [92], demonstrates that low-typed expressions have the same values in low-equivalent states.

Lemma 5 *If $\Gamma \vdash e : L$, then $\Gamma(x) = L$ for every variable x appearing in e. In particular, if $\Gamma \vdash e : L$ and $\sigma \sim_L \sigma'$, then $\sigma(e) = \sigma'(e)$.*

Proof: By induction on the structure of e. □

The following lemma demonstrates that configurations with high-typed commands take steps that preserve low-equivalence (in the sense that no low events are emitted and the resulting configuration is low-equivalent to the initial configuration).

Lemma 6 *If $\Gamma \vdash c : H$ cmd, then for all $\sigma, \psi, t,$ and ω, if*

$$(c, \sigma, \psi, t, \omega) \longrightarrow (c', \sigma', \psi', t', \omega'),$$

then $(c, \sigma, \psi, t, \omega) \sim_L (c', \sigma', \psi', t', \omega')$, and moreover $\Gamma \vdash c' : H$ cmd.

Proof: By induction on the derivation of $(c, \sigma, \psi, t, \omega) \longrightarrow (c', \sigma', \psi', t', \omega')$. □

The following lemma demonstrates that if the first command in a sequence terminates with some configuration, then the sequence eventually steps to an identical configuration with **skip** replaced by the second command in the sequence.

Lemma 7 *For all* $c_0, c_1, \sigma, \sigma', \psi, \psi', t, t', \omega,$ *and* ω'*, if*

$$(c_0, \sigma, \psi, t, \omega) \longrightarrow^* (\textsf{skip}, \sigma', \psi', t', \omega'),$$

then

$$(c_0; c_1, \sigma, \psi, t, \omega) \longrightarrow^* (c_1, \sigma', \psi', t', \omega').$$

Proof: By induction on the length of the derivation of

$$(c_0, \sigma, \psi, t, \omega) \longrightarrow^* (\textsf{skip}, \sigma', \psi', t', \omega'),$$

using rule SEQ-1 for the base case and rule SEQ-2 for the inductive case. □

The following lemma demonstrates that high-typed commands always terminate, and that the resulting terminal configuration is low-equivalent to the initial configuration.

Lemma 8 *If* $\Gamma \vdash c : H$ cmd*, then for any* σ, ψ, t *and* ω *there exists* σ', ψ', t' *and* ω' *such that*

$$(c, \sigma, \psi, t, \omega) \longrightarrow^* (\textsf{skip}, \sigma', \psi', t', \omega'),$$

and

$$(c, \sigma, \psi, t, \omega) \sim_L (\textsf{skip}, \sigma', \psi', t', \omega').$$

Proof: Note that a high-typed command cannot contain a **while**-statement. The result follows by structural induction on c, using Lemma 6 to demonstrate low equivalence for the base cases. For sequences we appeal to Lemma 7. □

The following lemma demonstrates that low-equivalent configurations with the same command take steps that preserve low equivalence.

Lemma 9 *For all* $c, \sigma_1, \sigma_2, \psi_1, \psi_2, t_1, t_2, \omega_1, \omega_2$, *and* m_1, *if*

$$(c, \sigma_1, \psi_1, t_1, \omega_1) \sim_L (c, \sigma_2, \psi_2, t_2, \omega_2), \text{ and } (c, \sigma_1, \psi_1, t_1, \omega_1) \longrightarrow m_1,$$

then there exists a configuration m_2 *such that*

$$(c, \sigma_2, \psi_2, t_2, \omega_2) \longrightarrow m_2, \text{ and } m_1 \sim_L m_2.$$

Proof: By induction on the derivation $(c, \sigma_1, \psi_1, t_1, \omega_1) \longrightarrow m_1$, using Lemma 5 for the rules Assign, Out, If-1, and If-2. □

The main nonprobabilistic lemma demonstrates that the traces emitted by low-equivalent configurations are low-equivalent.

Lemma 10 *For all configurations* m_1, m_2, *and* m_1', *if*

$$m_1 \sim_L m_2 \text{ and } m_1 \longrightarrow^* m_1',$$

then there exists a configuration m_2' *such that*

$$m_2 \longrightarrow^* m_2' \text{ and } m_1' \sim_L m_2'.$$

Proof: By induction on the length of the derivation of $m_1 \longrightarrow^* m_1'$. The base case is trivial. Otherwise, write $m_1 = (c_1, \sigma_1, \psi_1, t_1, \omega_1)$ and $m_2 = (c_2, \sigma_2, \psi_2, t_2, \omega_2)$, and consider the cases for $c_1 \sim_L c_2$:

(a) If $c_1 = c_2$, then suppose that $m_1 \longrightarrow m_1''$ and $m_1'' \longrightarrow^* m_1'$. By Lemma 9, there is a state m_2'' such that $m_2 \longrightarrow m_2''$ and $m_1'' \sim_L m_2''$. We can then apply the inductive hypothesis.

(b) If c_1 and c_2 are both high-typed, suppose that $m_1 \longrightarrow m_1''$ and $m_1'' \longrightarrow^* m_1'$. By Lemma 6, m_1'' is low equivalent to m_2, and we can apply the inductive hypothesis.

(c) If $c_1 = c_{H_1}; c$ and $c_2 = c_{H_2}; c$ for some command c and high-typed commands c_{H_1} and c_{H_2}, then consider the form of c_{H_1}. If $c_{H_1} = \textsf{skip}$, then $m_1 \longrightarrow \hat{m}$, where $\hat{m} = (c, \sigma_1, \psi_1, t_1, \omega_1)$, and since $(c, \sigma_1, \psi_1, t_1, \omega_1)$ is low equivalent to m_2, we can apply the inductive hypothesis. Otherwise, by Lemma 6 and SEQ-2, $m_1 \longrightarrow m_1''$ for some m_1'' such that is low equivalent to m_2, and we can apply the inductive hypothesis.

(d) If $c_1 = c_{H_1}; c_2$, then consider the form of c_{H_1}. If $c_{H_1} = \textsf{skip}$, then $m_1 \longrightarrow (c_2, \sigma_1, \psi_1, t_1, \omega_1)$, and since $(c_2, \sigma_1, \psi_1, t_1, \omega_1)$ is low equivalent to m_2, we can apply the inductive hypothesis. Otherwise, by Lemma 6 and SEQ-2, $m_1 \longrightarrow m_1''$ such that m_1'' is low equivalent to m_2, and we can apply the inductive hypothesis.

If $c_2 = c_{H_2}; c_1$, then by Lemma 8 and Lemma 7, there is a configuration $m_2'' = (c_1, \sigma_2'', \psi_2'', t_2'', \omega_2'')$ such that $m_2 \longrightarrow^* m_2''$ and $m_2 \sim_L m_2''$, and thus $m_1 \sim_L m_2''$. Suppose $m_1 \longrightarrow m_1''$ and $m_1'' \longrightarrow^* m_1'$. Then by Lemma 9, there is a configuration m_2''' such that $m_2'' \longrightarrow m_2'''$ such that $m_1'' \sim_L m_2'''$, and we can apply the inductive hypothesis.

□

The first two cases of Theorem 11 follow directly from this result.

C.2 Probabilistic proof details

We now generalize the results of the previous section to account for probabilistic programs. The structure of the proof is similar to the nonprobabilistic results.

Given a vertex v of a probability tree $\mathcal{T}_m$, let $\mathcal{T}_v$ denote the subtree of $\mathcal{T}_m$ rooted at v, and let $\mathcal{V}_v$ and $\mathcal{R}_v$ denote the sets of vertices and rays of $\mathcal{T}_v$, respectively. We denote the configuration with which v is labeled as $cf(v)$, and we write $v \sim_L v'$ if $cf(v) \sim_L cf(v')$.

Let a *frontier set* of a vertex v be a finite set of vertices $S \subseteq \mathcal{V}_v$ such that for every ray $\rho \in \mathcal{R}_v$ there exists exactly one vertex from S on ρ. Given a frontier set S of v, we call $F = (v, S)$ a *frontier*. Note that $\{v\}$ is a frontier set of v, and that given any frontier F we can obtain a new frontier F' by replacing any vertex in the frontier set with all of its descendants. Note also that a frontier set S partitions $\mathcal{R}_v$ into sets of rays that go through particular vertices in S.

Define the *depth* of a frontier (v, S) to be the length of the longest path (that is, the number of edges in the longest path) between v and vertices in S.

Because the vertices in a frontier $F = (v, S)$ induce a partition on the sets of rays going through v, the function π on vertices gives rise to a discrete probability measure on sets of vertices on S, normalized by the value of $\pi(v)$. More concretely, for any vertex v' in a frontier set S of v, let $\pi_v(v')$ be the product of the probabilities on the path from v to v'. We can now compare the distribution of low-equivalent configurations in two different frontiers. Given a frontier $F = (v, S)$ and a configuration m, let

$$[m]_F \quad \triangleq \quad \{v' \in S \mid cf(v') \sim_L m\}$$

be the subset of S whose configurations are low-equivalent to m. Define two

frontiers $F = (v, S)$ and $F' = (v', S')$ to be low-equivalent, denoted $F \sim_L F'$, if for any configuration m we have

$$\sum_{v'' \in [m]_F} \pi_v(v'') = \sum_{v'' \in [m]_{F'}} \pi_{v'}(v'').$$

The following lemma, which generalizes Lemma 7, demonstrates that if the first command in a sequence terminates in all execution paths, then the sequence eventually steps, in all execution paths, to the second command, while preserving other aspects of the original terminal configuration.

Lemma 11 *If v is a vertex in a probability tree such that $cf(v) = (c_0; c_1, \sigma, \psi, t, \omega)$, and $F_0 = (v_0, S_0)$ is a frontier such that*

- *$cf(v_0) = (c_0, \sigma, \psi, t, \omega)$,*
- *the subtree rooted at v_0 is finite, and*
- *S_0 consists of the root vertices of the subtree rooted at v_0,*

then there exists a frontier $F = (v, S)$ and a one-to-one mapping $g : S_0 \to S$ such that if $v'_0 \in S_0$ and $cf(v'_0) = (\mathsf{skip}, \sigma', \psi', t', \omega')$, we have $cf(g(v'_0)) = (c_1, \sigma', \psi', t', \omega')$.

Proof: By induction on the depth of F_0, using rules Seq-1 and Seq-2. □

The following lemma, which generalizes Lemma 8, demonstrates that high-typed commands terminate in all execution paths and that terminal configurations are low-equivalent to the initial configuration.

Lemma 12 *If v is a vertex in a probability tree such that $cf(v) = (c, \sigma, \psi, t, \omega)$ and $\Gamma \vdash c : H$ cmd, then the subtree rooted at v is finite and that for any leaf vertex v' of that subtree we have $v' \sim_L v$.*

Proof: By structural induction on c, using Lemma 11 for sequences and Lemma 6 to demonstrate low-equivalence for the base cases. □

The following lemma, generalizing Lemma 9, demonstrates that low-equivalent vertices have low-equivalent sets of children.

Lemma 13 *If $F_1 = (v_1, S_1)$ and $F_2 = (v_2, S_2)$ are frontiers such that*

- *$v_1 \sim_L v_2$,*
- *S_1 and S_2 are the sets of children of v_1 and v_2, and*
- *$cf(v_1)$ and $cf(v_2)$ share the same command c,*

then $F_1 \sim_L F_2$.

Proof: By structural induction on c, using Lemma 5 for the rules ASSIGN, OUT, IF-1, and IF-2. □

The following lemma is useful for the inductive cases of Lemma 15. It states that we can combine frontiers of vertices in low-equivalent frontiers to obtain deeper low-equivalent frontiers.

Lemma 14 *If $F_1 = (v_1, S_1)$ and $F_2 = (v_2, S_2)$ are frontiers such that*

- *$F_1 \sim_L F_2$;*
- *g_1 is a mapping from S_1 to frontier sets such that for any $v \in S_1$, $(v, g_1(v))$ is a frontier;*
- *g_2 is a mapping from S_2 to frontier sets such that for any $v \in S_2$, $(v, g_2(v))$ is a frontier; and*

- *for all* $v_1' \in S_1$ *and* $v_2' \in S_2$ *such that* $v_1' \sim_L v_2'$, *we have* $(v_1', g_1(v_1')) \sim_L (v_2', g_2(v_2'))$;

then $F_1' = (v_1, \cup_{v \in S_1} g_1(v))$ *and* $F_2' = (v_2, \cup_{v \in S_2} g_2(v))$ *are frontiers such that* $F_1' \sim_L F_2'$.

Proof: F_1' and F_2' are frontiers, which follows directly from the definition of a frontier set. To demonstrate the low-equivalence of F_1' and F_2', we must establish that for all configurations m we have

$$\sum_{v \in [m]_{F_1'}} \pi_{v_1}(v) = \sum_{v \in [m]_{F_2'}} \pi_{v_2}(v).$$

Let $\mathcal{M}$ denote a set of class representatives (for the equivalence relation $\sim_L$) of the set of configurations with which the elements of $S_1 \cup S_2$ are labeled. We have

$$\begin{aligned} \textstyle\sum_{v \in [m]_{F_1'}} \pi_{v_1}(v) &= \textstyle\sum_{v_1' \in F_1} \sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1}(v_1') \cdot \pi_{v_1'}(v) \\ &= \textstyle\sum_{m' \in \mathcal{M}} \sum_{v_1' \in [m']_{F_1}} \sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1}(v_1') \cdot \pi_{v_1'}(v) \\ &= \textstyle\sum_{m' \in \mathcal{M}} \sum_{v_1' \in [m']_{F_1}} \pi_{v_1}(v_1') \cdot \sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1'}(v). \end{aligned}$$

However, by assumption, for any configuration m and for any $v_1' \in S_1$ and $v_2' \in S_2$ such that $v_1' \sim_L v_2'$, we have $(v_1', g_1(v_1')) \sim_L (v_2', g_2(v_2'))$, and thus

$$\sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1'}(v) = \sum_{v \in [m]_{g_2(v_2')}} \pi_{v_2'}(v).$$

In fact, for any $w \in S_1$ such that $v_1' \sim_L w$, we also have

$$\sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1'}(v) = \sum_{v \in [m]_{g_1(w)}} \pi_w(v).$$

Thus, this sum depends only on the low-equivalence class of v_1'. We capture this by defining $s(m', m)$ to be equal to this sum, resulting in the following equalities:

$$\begin{aligned} s(m', m) &= \textstyle\sum_{v \in [m]_{g_1(v_1')}} \pi_{v_1'}(v) \qquad \text{for any } v_1' \in S_1 \text{ such that } v_1' \sim_L m' \\ &= \textstyle\sum_{v \in [m]_{g_2(v_2')}} \pi_{v_2'}(v) \qquad \text{for any } v_2' \in S_2 \text{ such that } v_2' \sim_L m'. \end{aligned}$$

We therefore have

$$\begin{aligned}
\textstyle\sum_{v\in[m]_{F_1'}} \pi_{v_1}(v) &= \textstyle\sum_{m'\in\mathcal{M}} \sum_{v_1'\in[m']_{F_1}} \pi_{v_1}(v_1')\cdot s(m',m) \\
&= \textstyle\sum_{m'\in\mathcal{M}} s(m',m)\cdot \sum_{v_1'\in[m']_{F_1}} \pi_{v_1}(v_1') \\
&= \textstyle\sum_{m'\in\mathcal{M}} s(m',m)\cdot \sum_{v_2'\in[m']_{F_2}} \pi_{v_2}(v_2') \qquad [\text{as } F_1 \sim_L F_2] \\
&= \textstyle\sum_{v\in[m]_{F_2'}} \pi_{v_2}(v), \qquad [\text{similarly}]
\end{aligned}$$

as desired. □

We can now state the main lemma, which generalizes Lemma 10.

Lemma 15 *If $F_1 = (v_1, S_1)$ is a frontier and $v_2 \sim_L v_1$, then there exists a frontier set S_2 of v_2 such that $F_1 \sim_L (v_2, S_2)$.*

Proof: By induction on the depth of F_1. The base case is trivial, and cases (a)–(d) are analogues of the cases in the nonprobabilistic proof. As before, write $cf(v_1) = (c_1, \sigma_1, \psi_1, t_1, \omega_1)$ and $cf(v_2) = (c_2, \sigma_2, \psi_2, t_2, \omega_2)$, and consider the cases for $c_1 \sim_L c_2$:

(a) Suppose that $c_1 = c_2$. If S_1' and S_2' are the sets of children of v_1 and v_2, we have $(v_1, S_1') \sim_L (v_2, S_2')$ by Lemma 13. The inductive hypothesis applies to $v_1' \in S_1'$ (with appropriate frontier sets $S_{v_1'} \subseteq S_1$) and elements $v_2' \in S_2'$ such that $v_1' \sim_L v_2'$, and the result follows by Lemma 14.

(b) If c_1 and c_2 are both high-typed, the result follows by Lemma 6, the inductive hypothesis applied to the children of v_1, and Lemma 14.

(c) If $c_1 = c_{H_1}; c$ and $c_2 = c_{H_2}; c$ for some command c and high-typed commands c_{H_1} and c_{H_2}, let S_1' be the set of children of v_1. For each $v_1' \in S_1'$ we have $v_1' \sim_L v_1 \sim_L v_2$ (by SEQ-1 if c_{H_1} is **skip**, or by Lemma 6 and SEQ-2 otherwise),

and we can therefore apply the inductive hypothesis. The result follows from Lemma 14.

(d) If $c_1 = c_{H_1}; c_2$, we can apply the inductive hypothesis using the same reasoning used for case (c). Otherwise we have $c_2 = c_{H_1}; c_1$. Let S_1' be the set of children of v_1. By Lemma 11 and Lemma 12, there exists a frontier set S_2' of v_2 such that for every element $v_2' \in S_2'$, v_2' is labeled with a configuration whose command is c_1, and $v_2' \sim_L v_2 \sim_L v_1$. Given any $v_2' \in S_2'$, let $S_{v_2'}$ be the set of children of v_2'. By Lemma 13 we have $(v_1, S_1') \sim_L (v_2', S_{v_2'})$, and the inductive hypothesis applies to elements $v_1' \in S_1'$ as in case (a). By Lemma 14 we can combine the frontiers of the elements of $S_{v_2'}$ to get a frontier $F_{v_2'} = (v_2', S_{v_2'})$ such that $F_{v_2'} \sim_L F_1$. Let $S_2 = \cup_{v_2' \in S_2'} S_{v_2'}$. We have $(v_2, S_2) \sim_L F_1$ by Lemma 14.

□

We are now ready to prove Theorem 11. We do so by establishing a connection between $\mu_m(\mathcal{E}_m(t))$, the probability that configuration m emits trace t, and the probabilities of vertices of arbitrarily deep frontiers of $\mathcal{T}_m$. Given a trace t and probability tree $\mathcal{T}_m$ with root vertex v_r and frontier (v_r, S), define:

$$\mathcal{E}_S(t) \triangleq \{\rho \in \mathcal{R}_m \mid \text{there exists a vertex } v \in S \text{ on } \rho \text{ and a trace } t' \text{ such that } tr(v) \text{ extends } t' \text{ and } t' \restriction L = t \restriction L\}.$$

The set $\mathcal{E}_S(t)$ consists of those rays on which traces that are low-equivalent to t appear at vertices that are ancestors of elements in S. Intuitively, $\mu_m(\mathcal{E}_S(t))$ is an approximation of $\mu_m(\mathcal{E}_m(t))$.

Theorem 11: *For any command c, if there exists a variable typing Γ and a security type τ such that $\Gamma \vdash c : \tau$ cmd, then*

(a) if c does not contain nondeterministic or probabilistic choice, then c satisfies noninterference;

(b) if c does not contain probabilistic choice, then c satisfies noninterference under refinement; and

(c) c satisfies probabilistic noninterference.

Proof: That c satisfies noninterference and noninterference under refinement follows from Lemma 10. To demonstrate that c satisfies probabilistic noninterference, we must show that, for all low-equivalent configurations m and m' and traces t, that $\mu_m(\mathcal{E}_m(t)) = \mu_{m'}(\mathcal{E}_{m'}(t))$. We demonstrate that $\mu_m(\mathcal{E}_m(t)) \leq \mu_{m'}(\mathcal{E}_{m'}(t))$; the reverse inequality is symmetric. Suppose, by way of contradiction, that $\mu_m(\mathcal{E}_m(t)) > \mu_{m'}(\mathcal{E}_{m'}(t))$. We demonstrate below that there exists a sequence of frontier sets $\{S_i\}$ of the root vertex v_r of $\mathcal{T}_m$ such that $\mu_m(\mathcal{E}_{S_i}(t))$ converges to $\mu_m(\mathcal{E}_m(t))$. It follows there exists a frontier set S of $\mathcal{T}_m$ such that $\mu_m(\mathcal{E}_S(t)) > \mu_{m'}(\mathcal{E}_{m'}(t))$. But by Lemma 15, there exists a frontier $F' = (v'_r, S')$ of $\mathcal{T}_{m'}$ such that $F' \sim_L (v_r, S)$. Thus $\mu_{m'}(\mathcal{E}_{S'}(t)) = \mu_m(\mathcal{E}_S(t))$, and $\mu_{m'}(\mathcal{E}_{S'}(t)) > \mu_{m'}(\mathcal{E}_{m'}(t))$. This is a contradiction, because $\mathcal{E}_{S'}(t) \subseteq \mathcal{E}_{m'}(t)$.

We now exhibit a sequence of frontier sets $\{S_i\}$ such that $\mu_m(\mathcal{E}_{S_i}(t))$ converges to $\mu_m(\mathcal{E}_m(t))$. Consider the sequence $S_0, S_1, \ldots, S_i, \ldots$ of frontier sets of v_r that comprise all the vertices at depth i of $\mathcal{T}_m$. We have $\mathcal{E}_m(t) = \cup_{i \geq 0} \mathcal{E}_{S_i}(t)$, and for all i we have $\mathcal{E}_{S_i}(t) \subseteq \mathcal{E}_{S_{i+1}}(t)$. Convergence follows due to a standard result in probability theory [3]. □

BIBLIOGRAPHY

[1] Luis von Ahn, A. Bortz, and Nicholas J. Hopper. k-anonymous message transmission. In *10th ACM Conference on Computer and Communications Security*, pages 122–130, 2003.

[2] Ana Almeida Matos, Gérard Boudol, and Ilaria Castellani. Typing noninterference for reactive programs. In *Proc. Workshop on Foundations of Computer Security*, 2004.

[3] Patrick Billingsley. *Probability and Measure*. Wiley-Interscience, 3rd edition, April 1995.

[4] Gérard Boudol and Ilaria Castellani. Noninterference for concurrent programs and thread systems. *Lecture Notes in Computer Science*, 281(1):109–130, 2002.

[5] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[6] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Towards privacy in public databases. In *Theory of Cryptography*, February 2005.

[7] Ellis Choen. Information transmission in computational systems. In *Proc. 6th ACM Symposium on Operating Systems Principles*, pages 133–139, November 1977.

[8] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–982, 1998.

[9] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. *Electronic Notes in Theoretical Computer Science*, 59(3), 2002.

[10] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. In *Proc. 18th IEEE Computer Security Foundations Workshop*, pages 31–45, June 2005.

[11] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Proc. Privacy Enhancing Technologies Workshop (PET 2003)*, volume 2760 of *Lecture Notes in Computer Science*, pages 54–68, Berlin/New York, 2003. Springer-Verlag.

[12] Bruno de Finetti. Les probabilités nulles. *Bulletins des Science Mathématiques (première partie)*, 60:275–288, 1936.

[13] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, 1977.

[14] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Approximate non-interference. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 3–17, 2002.

[15] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68, Berlin/New York, 2002. Springer-Verlag.

[16] E. Allen Emerson. Alternative semantics for temporal logics. *Theoretical Computer Science*, 26:121–130, 1983.

[17] Kai Engelhardt, Ron van der Meyden, and Yoram Moses. Knowledge and the logic of local propositions. In *Theoretical Aspects of Rationality and Knowledge: Proc. Seventh Conference (TARK 1998)*, pages 29–41, 1998.

[18] Alexandre Evfimievski, Johannes. E. Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. 22nd ACM Symposium on Principles of Database Systems*, pages 211–222, 2003.

[19] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, 1990.

[20] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995. A slightly revised paperback version was published in 2003.

[21] Riccardo Focardi and Roberto Gorrieri. A classification of security properties for process algebra. *Journal of Computer Security*, 3(1):5–33, 1994.

[22] Riccardo Focardi and Roberto Gorrieri. Classification of security properties (Part I: Information flow). In *Foundations of Security Analysis and Design*, pages 331–396. Springer, 2001.

[23] Riccardo Focardi and Sabina Rossi. Information flow security in dynamic contexts. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 307–319, 2002.

[24] Riccardo Focardi, Sabina Rossi, and Andrei Sabelfeld. Bridging language-based and process calculi security. In *Proc. of Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *LNCS*, April 2005.

[25] Cédric Fournet and Georges Gonthier. The Reflexive CHAM and the Join-Calculus. In *Conf. Record 23rd ACM Symposium on Principles of Programming Languages*, pages 372–385, 1996.

[26] Nir Friedman and Joseph Y. Halpern. Plausibility measures: a user's guide. In *Proc. Eleventh Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pages 175–184, 1995.

[27] Nir Friedman and Joseph Y. Halpern. Plausibility measures and default reasoning. *Journal of the ACM*, 48(4):648–685, 2001.

[28] Richard D. Gill, Mark J. van der Laan, and James M. Robins. Coarsening at random: Characterisations, conjectures and counter-examples. In *Proc. First Seattle Conference on Biostatistics*, pages 255–294, 1997.

[29] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, 1992.

[30] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical Report TR2003-1890, Cornell University Computing and Information Science, February 2003.

[31] Joseph A. Goguen and José Meseguer. Security policies and security models. In *Proc. IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[32] James W. Gray III and Paul F. Syverson. A logical approach to multilevel security of probabilistic systems. *Distributed Computing*, 11(2):73–90, 1998.

[33] Peter D. Grünwald and Joseph Y. Halpern. Updating probabilities. *Journal of A.I. Research*, 19:243–278, 2003.

[34] Joseph Y. Halpern. Conditional plausibility measures and Bayesian networks. *Journal of A.I. Research*, 14:359–389, 2001.

[35] Joseph Y. Halpern. Characterizing the common prior assumption. *Journal of Economic Theory*, 106(2):316–355, 2002.

[36] Joseph Y. Halpern. *Reasoning About Uncertainty*. MIT Press, Cambridge, Mass., 2003.

[37] Joseph Y. Halpern and Ronald Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–179, 1989.

[38] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[39] Joseph Y. Halpern and Kevin R. O'Neill. Secrecy in multiagent systems. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 32–46, 2002.

[40] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.

[41] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–514, 2005.

[42] Joseph Y. Halpern and Riccardo Pucella. Modeling adversaries in a logic for security protocol analysis. In *Proc. Formal Aspects of Security (FASec 2002)*, Lecture Notes in Computer Science, Volume 2629, pages 115–132. Springer-Verlag, Berlin/Heidelberg/New York, 2003.

[43] Joseph Y. Halpern and Riccardo Pucella. Probabilistic algorithmic knowledge. In *Theoretical Aspects of Rationality and Knowledge: Proc. Ninth Conference (TARK 2003)*, pages 118–130, 2003.

[44] Joseph Y. Halpern and Mark Tuttle. Knowledge, probability, and adversaries. *Journal of the ACM*, 40(4):917–962, 1993.

[45] Jifeng He, Karen Seidel, and Annabelle McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–192, 1997.

[46] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[47] Dominic Hughes and Vitaly Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[48] Sebastian Hunt and Dave Sands. On flow-sensitive security types. In *Conf. Record 33rd ACM Symposium on Principles of Programming Languages*, 2006.

[49] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic symbolic model checker. In P. Kemper, editor, *Proc. Tools Session of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pages 7–12, 2001. Available as Technical Report 760/2001, University of Dortmund.

[50] Henry E. Kyburg. Recent work in inductive logic. In T. Machan and K. Lucey, editors, *Recent Work in Philosophy*, pages 87–150. Rowman & Allanheld, Totowa, NJ, 1983.

[51] Brian N. Levine and Clay Shields. Hordes: A multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[52] Gavin Lowe. Quantifying information flow. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 18–31, 2002.

[53] Heiko Mantel. Possibilistic definitions of security: An assembly kit. In *Proc. 13th IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, United Kingdom, 2000.

[54] Heiko Mantel. *A uniform framework for the formal specification and verification of information flow security.* PhD thesis, Universität des Saarlandes, 2003.

[55] Heiko Mantel and Andrei Sabelfeld. A unifying approach to the security of distributed and multi-threaded programs. *Journal of Computer Security*, 11(4):615–676, September 2003.

[56] Daryl McCullough. Specifications for multi-level security and a hook-up property. In *Proc. IEEE Symposium on Security and Privacy*, pages 161–166, 1987.

[57] John McLean. Proving noninterference and functional correctness using traces. *Journal of Computer Security*, 1(1):37–58, 1992.

[58] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symposium on Security and Privacy*, pages 79–93, 1994.

[59] Ron van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[60] Ron van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. 17th IEEE Computer Security Foundations Workshop*, pages 280–291, 2004.

[61] Robin Milner. Processes: A mathematical model of computing agents. In H. E. Rose and J. C. Shepherdson, editors, *Proceedings of the Logic Colloquium, Bristol, July 1973*, pages 157–173, New York, 1975. American Elsevier Pub. Co.

[62] Robin Milner. *A Calculus of Communicating Systems.* Lecture Notes in Computer Science, Volume 92. Springer-Verlag, Berlin/New York, 1980.

[63] John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols, 2006. To appear, *Theoretical Computer Science.*

[64] Stephen Morris. The common prior assumption in economic theory. *Economics and Philosophy*, 11:227–253, 1995.

[65] Andrew C. Myers, Andrei Sabelfeld, and Steve Zdancewic. Enforcing robust declassification. In *Proc. 17th IEEE Computer Security Foundations Workshop*, pages 172–186, 2004.

[66] Andrew C. Myers, Lantian Zheng, Steve Zdancewic, Stephen Chong, and Nathaniel Nystrom. Jif: Java information flow. Software release. Located at `http://www.cs.cornell.edu/jif`, 2001–2005.

[67] Kevin R. O'Neill, Michael R. Clarkson, and Stephen Chong. Information-flow security for interactive programs. Technical Report TR2006-2022, Cornell University, Ithaca, NY, Apr 2006.

[68] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudeonymity: a proposal for terminology. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 1–9, New York, 2001. Springer-Verlag.

[69] Karl R. Popper. *The Logic of Scientific Discovery*. Hutchison, London, 2nd edition, 1968. The first version of this book appeared as *Logik der Forschung*, 1934.

[70] Michael O. Rabin. n-process mutual exclusion with bounded waiting by $4 \cdot \log n$-valued shared variable. *Journal of Computer and System Sciences*, 25(1):66–75, 1982.

[71] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[72] A. W. Roscoe. CSP and determinism in security modeling. In *Proc. IEEE Symposium on Security and Privacy*, 1995.

[73] Peter Y. A. Ryan and Steve A. Schneider. Process algebra and non-interference. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 214–227, 1999.

[74] Peter Y. A. Ryan, Steve A. Schneider, Michael H. Goldsmith, Gavin Lowe, and A. W. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, Harlow, England, 2001.

[75] Andrei Sabelfeld and Heiko Mantel. Static confidentiality enforcement for distributed programs. In *Proceedings of the 9th International Static Analysis Symposium*, volume 2477 of *LNCS*, Madrid, Spain, September 2002. Springer-Verlag.

[76] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[77] Andrei Sabelfeld and David Sands. Probabilistic noninterference for multi-threaded programs. In *Proc. 13th IEEE Computer Security Foundations Workshop*, pages 200–214. IEEE Computer Society Press, July 2000.

[78] Steve A. Schneider. Security Properties and CSP. In *Proc. 1996 IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

[79] Steve A. Schneider and Abraham Sidiropoulos. CSP and anonymity. In *European Symposium on Research in Computer Security*, pages 198–218, 1996.

[80] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul F. Syverson, editors, *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53, Berlin/New York, 2002. Springer-Verlag.

[81] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28-4:656–715, 1949.

[82] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy*, pages 58–70, 2002.

[83] Vitaly Shmatikov. Probabilistic analysis of anonymity. In *Proc. 15th Computer Security Foundations Workshop*, pages 119–128, 2002.

[84] Vincent Simonet. Fine-grained information flow analysis for a lambda-calculus with sum types. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 223–237, June 2002.

[85] Vincent Simonet. The Flow Caml System: Documentation and user's manual. Technical Report 0282, Institut National de Recherche en Informatique et en Automatique (INRIA), July 2003.

[86] Geoffrey Smith. A new type system for secure information flow. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 115–125, 2001.

[87] David Sutherland. A model of information. In *Proc. 9th National Security Conference*, pages 175–183, 1986.

[88] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, 1997.

[89] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods*, pages 814–833, 1999.

[90] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–338, 1985.

[91] Dennis Volpano and Geoffrey Smith. Probabilistic noninterference in a concurrent language. *Journal of Computer Security*, 7(2,3):231–253, November 1999.

[92] Dennis Volpano, Geoffrey Smith, and Cynthia Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.

[93] J. Todd Wittbold and Dale M. Johnson. Information flow in nondeterministic systems. In *Proc. IEEE Symposium on Security and Privacy*, pages 144–161, May 1990.

[94] Aris Zakinthinos and E. S. Lee. A general theory of security properties. In *Proc. IEEE Symposium on Security and Privacy*, pages 94–102, 1997.

[95] Steve Zdancewic and Andrew C. Myers. Robust declassification. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 15–23, 2001.

[96] Steve Zdancewic and Andrew C. Myers. Observational determinism for concurrent program security. In *Proc. 16th IEEE Computer Security Foundations Workshop*, pages 29–43, Pacific Grove, California, June 2003.